LIBRO DIGITALE
eBook + Libro

Marisa Addomine Daniele Pons

Informatica

Reti di comunicazione, princìpi di computazione, fondamenti di calcolo numerico



ZANICHELLI

Marisa Addomine
Daniele Pons

Informatica

Reti di comunicazione,
princìpi di computazione,
fondamenti di calcolo numerico

Realizzazione editoriale:
- Coordinamento editoriale: Matteo Fornesi
- Redazione, composizione, impaginazione, illustrazioni e ricerca iconografica: Studio Monza s.a.s., Milano
- Segreteria di redazione: Deborah Lorenzini
- Progetto grafico: Editta Gelsomini

Gli autori ringraziano per la collaborazione il professor Francesco Colosi

Copertina:
- Progetto grafico: Miguel Sal & C., Bologna
- Realizzazione: Roberto Marchetti
- Immagini di copertina: Seamartini Graphics/Shutterstock, ConstantinosZ/Shutterstock

Prima edizione: marzo 2014

Marisa Addomine
Daniele Pons

Informatica

Reti di comunicazione,
princìpi di computazione,
fondamenti di calcolo numerico

ZANICHELLI

Sommario

Help

Il testo

I contenuti di questo volume sono strutturati in sezioni e capitoli e sono corredati da schede di approfondimento. Al termine di ogni capitolo trovate un riepilogo dei concetti essenziali trattati e una serie di test, di esercizi e di problemi da risolvere. Nella pagina a lato trovate una descrizione più dettagliata.
All'interno del testo scritto sono state utilizzate alcune convenzioni (codici di lettura) per segnalare diverse valenze del testo stesso.

Codici di lettura

Alcuni elementi del testo sono messi in evidenza da caratteri e coilori diversi, per permettere di riconoscere parole chiave, concetti importanti o termini tecnici dei linguaggi e delle applicazioni. Eccoli:

numerazione binaria

Questo carattere è usato per indicare che la voce così formattata è inserita nel Glossario, che potete trovare sul sito Web associato al presente volume. Le voci di glossario sono tipiche del mondo informatico e ne vengono riportati il significato e l'eventuale traduzione.

Inserisci > Immagine > Da File

Una notazione così formattata mostra un comando (o una sequenza di comandi) da eseguire richiamandolo dai menu dell'applicazione.

```
char linea_pari[LINEA];
```

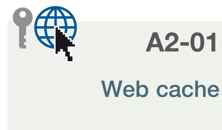
Un testo così formattato evidenzia programmi o porzioni di programmi, come vengono scritti in fase di creazione del codice.

strcat(<var1>, <var2>)

Questa formattazione del testo mostra la sintassi con cui, nei linguaggi di programmazione, si scrivono le istruzioni.

Rimandi al sito Web

Alcuni approfondimenti dei temi trattati nel testo sono disponibili online sul sito del corso.
La loro presenza è segnalata, nelle pagine del testo, da una apposita grafica. Qui di seguito ne è riportato un esempio:



Se nell'icona è indicata una chiave, l'accesso è protetto da password.

Sezioni e capitoli

Una sigla composta da una lettera e da un numero indica la sezione del testo e il capitolo in cui ci troviamo.

Introduzione

Le note introduttive presentano il tema del capitolo, il contesto storico, teorico o pratico e gli obiettivi didattici.

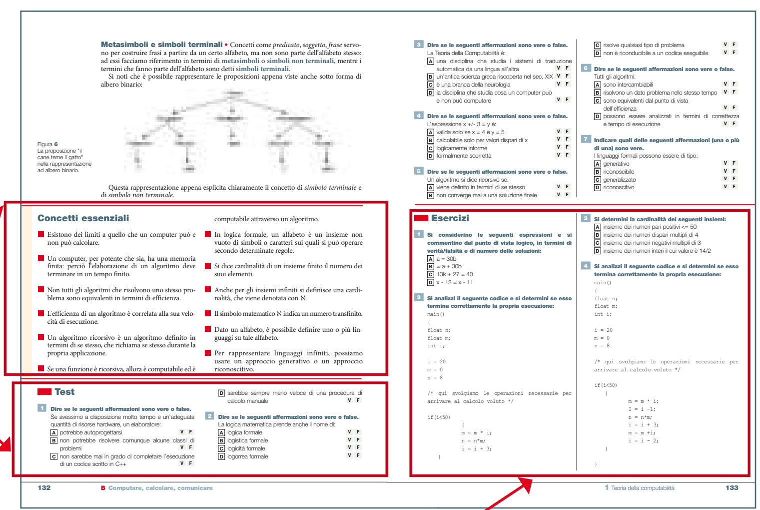


Schede di approfondimento

Contengono informazione aggiuntiva rispetto al testo principale o ampliano temi di particolare importanza, sia teorica che pratica.

Concetti essenziali

Vengono ripresi sinteticamente i punti fondamentali del capitolo: puntualizzano i capisaldi di quanto appreso e ne permettono una verifica veloce.



Test

Per ogni capitolo, una serie di test a scelta multipla ci permette di verificare l'apprendimento dei concetti fondamentali e della corretta terminologia.

Esercizi

Rappresentano un momento applicativo in cui è possibile verificare la capacità di risolvere problemi specifici grazie a quanto appreso nel capitolo.

Sezione **A**

Reti di comunicazione

1 Dal generale al particolare

La facilità di accesso alle risorse di rete rende disponibile un universo di servizi del tutto inimmaginabili sino a qualche decennio fa. Il capitolo riparte da alcuni concetti introdotti negli anni precedenti, prendendo spunto dalla struttura dei diversi tipi di rete, tra cui anche la rete cellulare e quella WiFi, distinguendo tra l'aspetto logico e quello fisico, e mostrando potenzialità e limiti delle diverse opzioni di scelta a disposizione del progettista e del sistemista.

Sono quindi presentati concetti fondamentali, quali il ritardo nodale, il traffico di rete, i protocolli e i modelli di servizio e il principio fondamentale dell'incapsulamento.

1 Cos'è l'Internet

Negli anni scorsi, abbiamo definito una rete di computer come ***un insieme di computer e periferiche, collegati da canali di comunicazione che permettono la condivisione di dati e risorse, sia hardware che software***. Abbiamo anche accennato al suo funzionamento, alla suddivisione in vari livelli ed al modello **ISO/OSI** che permette l'interoperabilità di tutti i componenti della rete stessa.

Abbiamo visto come le reti vengano, per comodità, classificate secondo la loro funzione ed estensione, dalla piccola rete locale casalinga alle grandi dorsali di comunicazione intercontinentali, e come l'Internet possa essere definita ***la rete delle reti***, dato che utilizza tutti i livelli disponibili per interconnettere qualunque nodo.

È ovvio che una piccola **LAN** casalinga non presenta la stessa struttura della rete geografica dei grandi **carrier**, ma i mattoni fondamentali che compongono entrambe sono gli stessi, anche se cambiano prestazioni e interfacce, e identici sono anche i linguaggi e i **protocolli** che utilizzano. Per questa ragione, seguiremo un approccio descrittivo che va dal generale al particolare, partendo dal livello più alto e scendendo sempre più in profondità.

Due punti di vista • Per prima cosa, dobbiamo definire con maggiore esattezza cos'è l'Internet. Per farlo, possiamo utilizzare due approcci diversi, entrambi necessari per averne una visione globale.

Il primo consiste nel descrivere i mattoni fondamentali, sia hardware che software, che la compongono, ossia la sua struttura fisica; il secondo permette di descriverla come un'infrastruttura che fornisce servizi ad applicazioni distribuite, vale a dire in termini della sua struttura logica.

Struttura fisica • Se guardiamo l'Internet dal primo punto di vista, si tratta effettivamente di una rete che interconnette centinaia di milioni di apparati distribuiti su tutto il pianeta.

Il temine "*rete di computer*", tuttavia, è ormai obsoleto e praticamente inapplicabile. Sino a non molto tempo fa, infatti, questi apparati consistevano effettivamente in PC, workstation e server, che immagazzinavano e si scambiavano pagine Web, e-mail e altre informazioni dello stesso tipo; negli ultimi anni, però, un numero sempre crescente - e ormai maggioritario - di apparati si connette all'Internet: smartphone e tablet, telefoni, console per videogiochi e televisori, automobili e navigatori satellitari, webcam e stazioni meteorologiche, sistemi di

sorveglianza e sistemi di **domotica**, tanto per citarne alcuni. Nel gergo dell'Internet, tutti questi apparati sono chiamati **host** oppure **end system** (traducibili come *nodo ospite*).

I nodi sono connessi tra loro da **communication link**, ossia collegamenti fisici, e da **packet switch** (in italiano *commutatori di pacchetti*): vedremo più avanti le tipologie e le particolarità di entrambi.

Quando un nodo ospite deve inviare dati a un altro, li segmenta in blocchi ed aggiunge un h**eader** ad ogni segmento: i blocchi così ottenuti sono chiamati **packet** (*pacchetti)*. I pacchetti sono inviati sulla rete e instradati dai *packet switch* sino al dispositivo destinatario, che li apre e li riassembla in modo da ricostruire il messaggio originario.

Sul percorso tra i due nodi si trova un numero non conoscibile a priori di *packet switch*, ognuno dei quali riceve i pacchetti su una delle sue porte di ricezione, e li reinstrada su una delle sue porte di trasmissione, seguendo l'indirizzo contenuto negli *header* sino a raggiungere la destinazione finale. Esistono diversi tipi di dispositivi di commutazione e instradamento riuniti nella grande famiglia dei *packet switch*, ma attualmente la struttura dell'Internet è composta principalmente di **router** e **switch di layer 2**, dispositivi di cui parleremo in dettaglio più avanti.

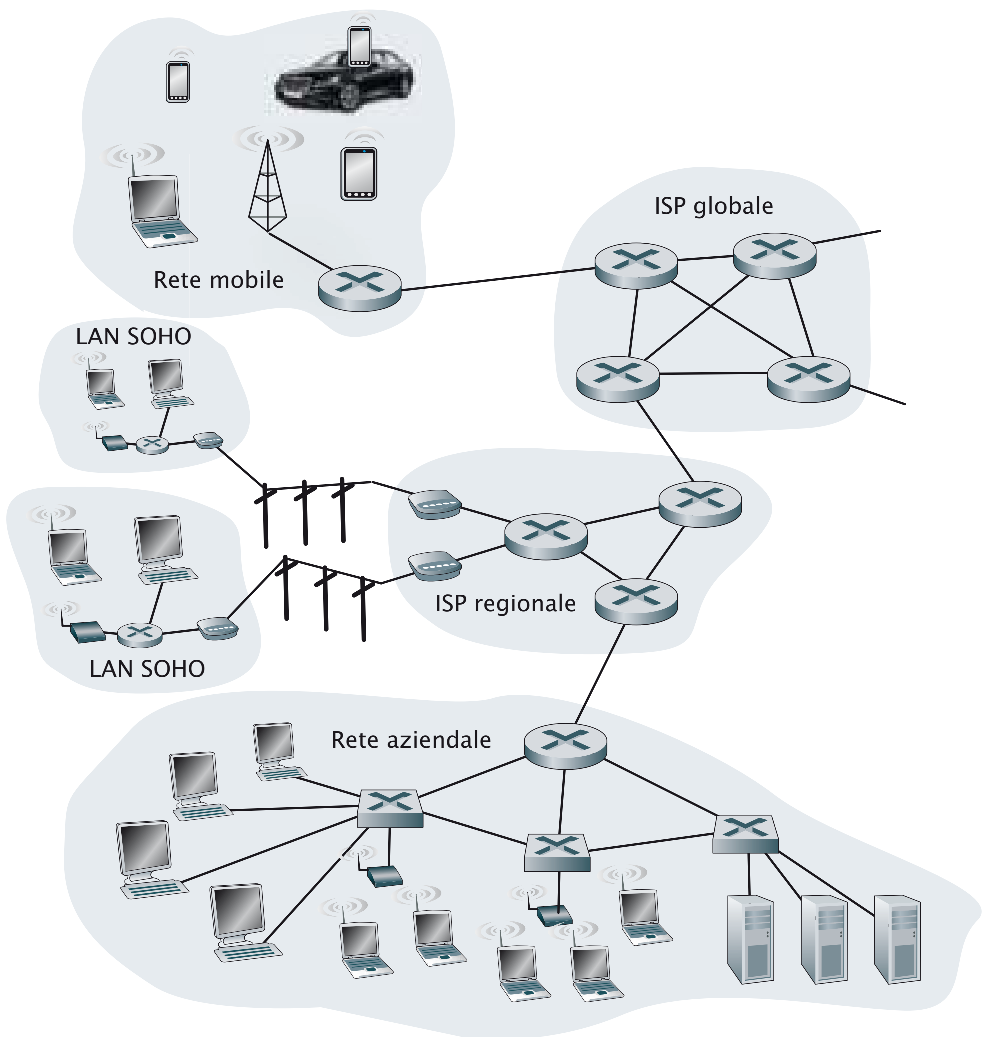


Figura **1**
Struttura generale dell'Internet.

Una struttura di questo tipo è assimilabile alla rete delle vie di trasporto, formata da grandi dorsali, intersezioni, strade statali, comunali, sino ai piccoli raccordi delle singole abitazioni.

Immaginiamo che una azienda debba spostare un grosso lotto di materiale dalla propria sede fino a un magazzino molto distante. I materiali saranno suddivisi su un certo numero di autocarri, ciascuno dei quali, uscito dalla fabbrica, si immetterà su una strada locale, entrerà poi su una strada di maggior traffico e da lì, attraverso un'intersezione, su un'autostrada, per poi uscirne - sfruttando un'altra intersezione - e prendere strade sempre meno importanti, sino alla destinazione.

Se consideriamo gli autocarri come pacchetti, le strade come *communication link* e le intersezioni come *switch*, abbiamo ottenuto una buona immagine del funzionamento dell'Internet: attraverso una sequenza di percorsi e instradamenti, i pacchetti viaggiano dal mittente al destinatario, immettendosi in strade sempre più grandi e trafficate, per poi uscirne e percorrerne di sempre minori, sino all'arrivo.

Gli *host* accedono all'Internet tramite gli **ISP** (*Internet Service Providers*), che forniscono diverse modalità di accesso e sono essi stessi delle reti.

Sono strutturati a diversi livelli gerarchici: si va dagli ISP locali, ad esempio le reti WiFi degli aeroporti, fino ai grandi *carrier* nazionali e internazionali, quali, ad esempio, Telecom, Vodafone, AT&T. Più avanti, esamineremo in maggior dettaglio gli ISP e le loro caratteristiche.

Struttura logica • Se consideriamo l'Internet come un'infrastruttura che fornisce servizi, notiamo subito la grande varietà di applicazioni che ad essa si appoggiano: posta elettronica, navigazione Web, social networking, messaggistica istantanea, **VoIP**, streaming video, giochi online, condivisione di file **P2P** (*Peer to Peer*), accesso remoto sono solo le più comuni.

È importante notare come le applicazioni siano residenti solo sui nodi ospiti, e non girino in alcun modo sui componenti dell'infrastruttura, i quali si limitano a veicolare i dati prodotti e non hanno alcun ruolo nella loro generazione e nella loro lettura.

Diventa, quindi, fondamentale, la capacità delle applicazioni di veicolare correttamente i dati sull'Internet, ossia di istruire l'infrastruttura su come raggiungere i destinatari.

Gli *host* connessi all'Internet dispongono di un'**API** (*Application Programming Interface*), che specifica le modalità con cui le applicazioni chiedono all'infrastruttura di consegnare i dati ad una specifica destinazione: si tratta di un insieme di regole che l'applicazione che spedisce deve seguire affinché l'Internet comprenda e individui l'*host* destinatario.

Riprendendo l'analogia con il sistema postale già utilizzata gli anni scorsi, supponiamo di voler inviare una lettera a un'altra persona. Il testo della lettera corrisponde ai dati da inviare, ma occorre una busta con l'indirizzo e l'affrancatura perché il servizio postale sia in grado di effettuare la consegna. L'indirizzo sulla busta, poi, deve essere scritto secondo un criterio prestabilito: l'indirizzo del mittente in alto a sinistra, quello del destinatario in basso a destra e l'affrancatura in alto a destra; l'indirizzo del destinatario, inoltre, deve avere un formato preciso: prima riga nome e cognome, seconda riga nome e numero della via, terza riga codice postale, località, comune e sigla della provincia, messa tra parentesi.

Si può dire che queste regole rappresentino l'API del servizio postale.

Il servizio postale, però, oltre alla consegna della corrispondenza fornisce molti altri servizi, ognuno dei quali ha regole diverse e, spesso, richiede la compilazione di moduli prestampati da compilare per poterne usufruire. Allo stesso modo, l'Internet fornisce un gran numero di servizi alle applicazioni, con regole di accesso diverse e, quindi, diverse modalità di comunicazione. Vedremo nel capitolo successivo quali sono i principali servizi forniti dall'Internet e le loro caratteristiche.

Restando nell'analogia, notiamo come le regole che definiscono **cosa** va scritto sulla busta e sugli eventuali moduli aggiuntivi siano accompagnate da altre che indicano **come** queste istruzioni debbano essere fornite: la sequenza, la posizione, l'eventuale richiesta di ricevuta e, implicitamente, l'alfabeto da utilizzare.

Queste ulteriori regole sono i *protocolli di comunicazione*.

Riprendendo quanto già scritto, possiamo dire che con il termine *protocollo di rete* si intende la definizione formale delle regole che gli *host* collegati tra loro devono rispettare per

instaurare in modo efficace la comunicazione. Poiché una rete è strutturata su diversi strati, che operano funzioni differenti, ogni strato richiede la definizione di un proprio protocollo, che deve essere comune a tutti i software che operano a quel livello.

In sintesi, ***un protocollo definisce il formato e l'ordine dei messaggi scambiati tra due o più apparati connessi in rete, così come le azioni che devono essere intraprese per trasmettere e ricevere un messaggio o un altro evento***.

2 Network Edge - reti di accesso e mezzo fisico

Dopo aver fatto una panoramica generale dell'Internet, scendiamo più in profondità nella struttura e nei componenti di una rete complessa, iniziando dalla sua periferia, ossia dalle **reti di accesso** (in inglese *Access Networks*), che garantiscono la connessione degli *host* al sistema.

Gli *host* vengono comunemente divisi in *client* e *server*: in modo informale, possiamo dire che i primi sono gli apparati su cui girano le applicazioni utilizzate dagli utenti finali, i secondi sono i computer che contengono le applicazioni che gestiscono le informazioni, la loro conservazione e la loro distribuzione. In realtà, in alcuni casi, come lo scambio di file, client e server si fondono nello stesso dispositivo.

Chiamiamo rete di accesso quella che connette fisicamente un host al primo router dell'Internet, chiamato perciò *edge router*.

Esistono diversi tipi di reti di accesso, diverse per funzione e collocazione fisica: reti casalinghe e di piccole aziende (dette **SOHO**, dall'inglese *Small Office Home Office*), grandi reti aziendali, reti mobili, a loro volta implementate con tecnologie e prestazioni diverse.

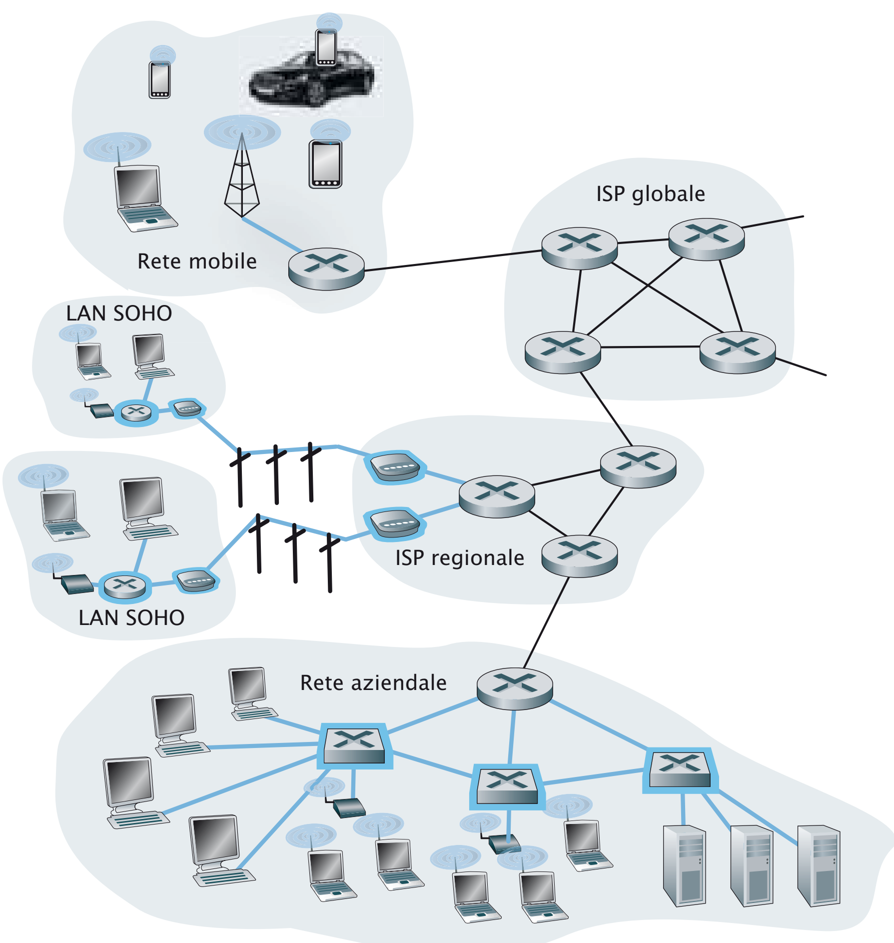


Figura **2**
Reti di accesso.

Dal punto di vista delle connessioni, troviamo reti di accesso che utilizzano cavo in rame, fibra ottica e radio. Queste ultime, a seconda delle tecnologie usate, coprono aree di diversa estensione: si passa dalle poche decine di metri delle **WLAN** alla copertura cellulare del territorio per i terminali mobili, fino a sistemi, terrestri e satellitari, che consentono la connessione all'Internet di apparati fissi anche dove non è possibile far giungere un adeguato collegamento fisico.

Le reti di accesso SOHO rappresentano la maggioranza delle reti di accesso, dato che ormai, nei Paesi sviluppati, ne dispongono quasi l'80% delle abitazioni e praticamente tutti gli uffici e le piccole aziende.

Nella maggior parte dei casi, la connessione avviene utilizzando la linea telefonica o, più raramente, una linea in fibra ottica; negli Stati Uniti è molto sfruttato anche il cavo coassiale della TV via cavo, che in Europa praticamente non esiste.

Connessione via cavo • La connessione tramite la linea telefonica è possibile grazie alla tecnologia **DSL** (*Digital Subscriber Line*), che permette di trasportare segnali digitali ad alta velocità sul normale doppino telefonico, pur con limitazioni dovute alla distanza tra la centrale e l'utente finale.

Il segnale digitale che deve uscire dalla rete viene trasformato in un segnale analogico sinusoidale da un modem che, attraverso uno **splitter**, cioè un *separatore*, lo inietta nel doppino telefonico. All'altra estremità del doppino, nella centrale telefonica, si trova un apparato chiamato **DSLAM** (*Digital Subscriber Line Access Multiplexer*) che separa il segnale da quello telefonico e lo digitalizza, per poi instradarlo verso il router di accesso.

Allo stesso modo, il segnale digitale che proviene dal router di accesso ed è destinato ad un utente viene prelevato dal DSLAM, trasformato in segnale analogico e inviato sulla linea telefonica sino al modem del destinatario, che provvede a digitalizzarlo di nuovo. Il flusso dei dati dall'utente finale alla centrale viene detto **upstream**, quello inverso **downstream**.

È possibile inviare contemporaneamente sul doppino sia le chiamate telefoniche, sia la comunicazione digitale, perché vengono utilizzate bande di frequenza diverse, che non interferiscono tra loro:

- il canale telefonico con banda sino a 4 kHz
- il canale di upstream con banda tra 4 kHz e 50 kHz
- il canale di downstream con banda tra 50 kHz e 1 MHz

Appare ovvio che il flusso di downstream è molto più veloce di quello di upstream, disponendo di larghezza di banda molto superiore: in effetti, i doppini esistenti sono stati installati quando questa tecnologia non esisteva, e le loro caratteristiche richiedono di scendere a compromessi per quanto riguarda le prestazioni.

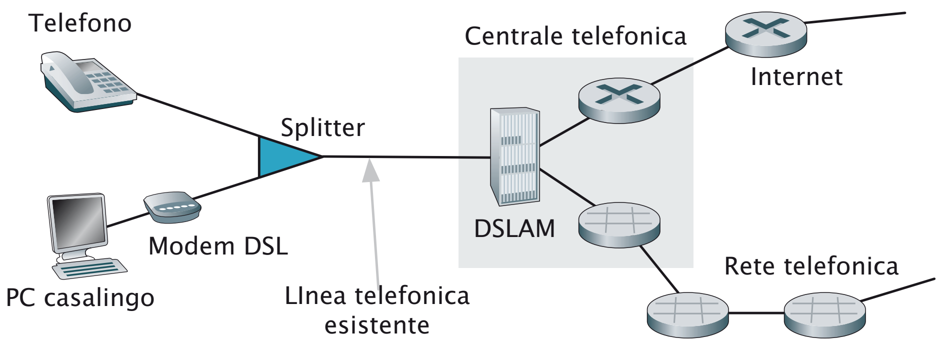


Figura **3**
Accesso tramite DSL.

ADSL • Per questa ragione, è stata adottata una particolare versione del DSL, detta asimmetrica per questa sua caratteristica, e quindi chiamata **ADSL**.

La crescente velocità di connessione pubblicizzata dai carrier telefonici che fungono anche da ISP non è dovuta ad un aumento della larghezza di banda disponibile sul doppino, ma alla realizzazione di chip **DSP** (*Digital Signal Processing*) sempre più raffinati e potenti, che permettono ai modem e ai DSLAM di inviare e ricevere una quantità di dati sempre maggiore, a parità di banda: attualmente, le offerte degli ISP prevedono connessioni sino a 10 Mbps reali.

Connessione via fibra ottica • Si va sempre più estendendo la posa di fibra ottica, oltre che sulle dorsali, anche anche sul cosiddetto ultimo miglio, ossia il tratto fra l'utente finale e la centrale più vicina: ciò consente di superare le limitazioni di banda insite nell'uso del doppino telefonico.

Esistono diverse modalità di connessione tramite fibra ottica, che dipendono dall'infrastruttura disponibile, dalle prestazioni che si intendono raggiungere e dal costo che l'utente finale è disposto a pagare.

In ogni caso, possiamo distinguere tra due architetture fondamentali: la prima, detta **FTTS** (*Fiber To The Street*) prevede la stesura di fibra ottica sino a distributori locali, detti in gergo *cabinet*, da cui parte un normale doppino, di lunghezza non superiore ai 250 m, che raggiunge l'utente finale; la seconda, detta **FTTH** (*Fiber To The Home*), copre con la fibra l'intera tratta fra la centrale e l'utente finale.

FTTS • L'FTTS garantisce prestazioni superiori a quelle ottenibili con l'ADSL via doppino telefonico, ma comporta in ogni caso un tratto in cui il segnale è analogico modulato e quindi non consente la simmetria nella velocità di connessione, utilizzando ancora la tecnologia DSL.
Questa architettura è ampiamente usata dai carrier telefonici nelle aree cittadine, dove la posa di fibra ottica non deve coprire distanze elevate e, quindi, non richiede eccessivi investimenti; al contempo, non sono richieste modifiche alle connessioni all'interno degli edifici dove si trovano gli utenti finali.
Dal punto di vista della struttura, l'utente finale continua a disporre di un modem, mentre modem e DSLAM sono sostituiti da router con convertitori rame-fibra e modem integrati.

FTTH • L'FTTH può essere realizzato in diversi modi, che dipendono dalle tecnologie scelte dal carrier per la realizzazione della rete distributiva.
Il sistema più semplice prevede la connessione diretta dell'utente finale alla centrale tramite un convertitore chiamato **ONT** (*Optical Network Terminator*), che converte il segnale da elettrico a ottico, la cui uscita viene connessa ad un ripartitore, che raccoglie un gruppo di fibre ottiche in una sola, collegata direttamente alla centrale.
Qui un ulteriore apparato, chiamato **OLT** (*Optical Line Terminator*), opera la conversione del segnale da ottico a elettrico e lo invia all'*edge router*.
Questa soluzione è la più semplice anche dal punto di vista impiantistico e, anche se non ha molti dei vincoli dell'ADSL, presenta comunque una velocità di trasferimento limitata dal numero di utenti connessi ad un singolo splitter, attualmente limitata a circa 20 Mbps effettivi.

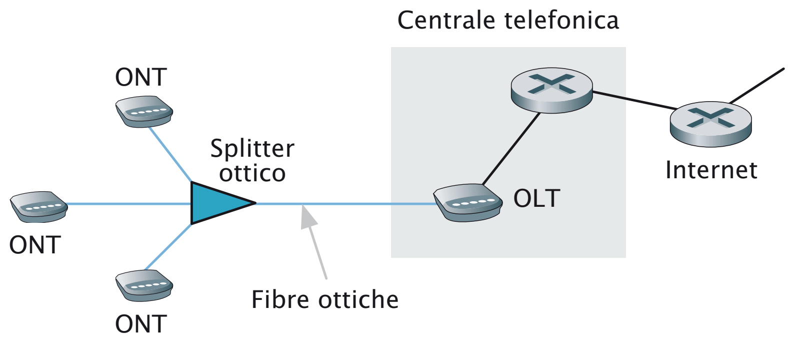


Figura **4**
Accesso tramite fibra ottica.

Grandi LAN • Molto più raffinata e performante è la soluzione scelta da alcuni carrier, in particolare MetroWeb in Italia, di realizzare reti metropolitane strutturandole come una grande LAN ed evitando, quindi, qualsiasi apparato terminale tra la rete dell'utente finale e la propria.

In questo modo, l'*edge router* verso l'Internet è spostato molto lontano rispetto alla periferia, ma poiché il trasporto dei dati avviene senza interposizione di modem e convertitori, si raggiungono elevatissime velocità di connessione (sino a 100 Mbps effettivi).

Fisicamente, la LAN dell'utente finale viene connessa a un router, che a sua volta è connesso ad uno switch che fa parte della gigantesca LAN metropolitana la quale, in qualche suo punto, converge su uno o più *edge router* molto potenti, che gestiscono la connessione all'Internet dell'intero sistema. Questa tecnologia comporta un particolare utilizzo dei **NAT** (*Network Address Translator*), di cui parleremo nel capitolo 3.

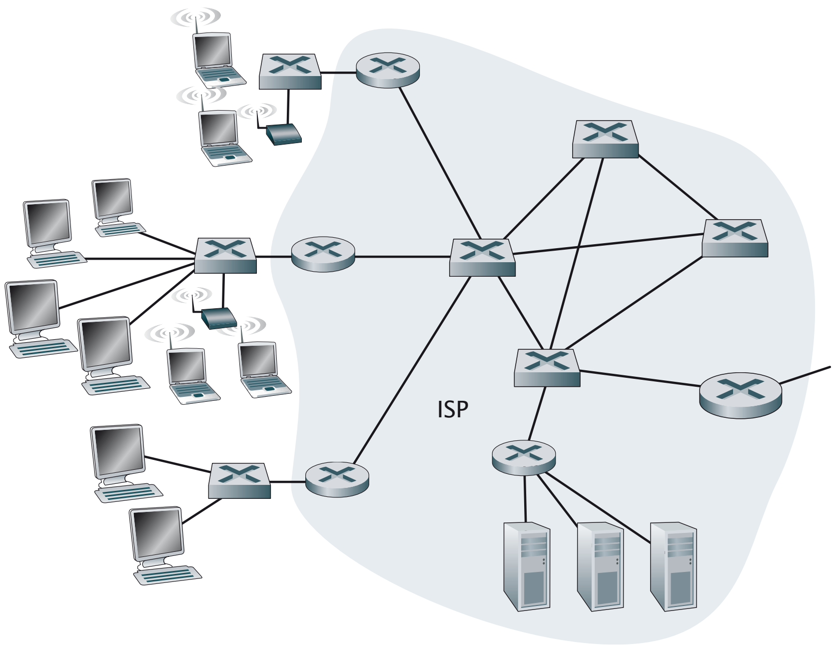


Figura **5**
Rete metropolitana con struttura LAN.

Connessione via radio • Come abbiamo già accennato, esistono reti via radio che adottano varie tecnologie, coprono aree molto diverse e hanno differenti funzioni.

Ci limitiamo a citare le connessioni satellitari, che vengono utilizzate in situazioni estreme, ad esempio le comunicazioni con l'Antartide, oppure quando grandi organizzazioni vogliono connettere tra loro più sedi senza transitare da infrastrutture pubbliche.

Cellulari • Molto più esteso e sempre più importante è l'accesso all'Internet tramite le reti cellulari: questo tipo di connessione interessa un numero sempre più esteso e diversificato di dispositivi mobili. Un elenco non esaustivo comprende smartphone, tablet, laptop, navigatori satellitari, sistemi di infotainment e antifurto delle automobili, sistemi di rilevazione meteorologici e di inquinamento, sistemi di controllo di strutture distribuite (acquedotti, gasdotti, centrali idroelettriche), sistemi di controllo di infrastrutture (ponti, dighe, tunnel), sistemi di rilevazione del traffico, sistemi di comunicazione con mezzi di trasporto pubblici, quali treni e autobus.

Già negli ultimi anni del secolo scorso erano in opera strutture di reti cellulari basate sullo standard *GSM*, che permetteva la trasmissione di dati a velocità che oggi ci apparirebbero piuttosto modeste.

La sempre maggiore richiesta di mobilità nelle connessioni Internet ha spinto le **Telco** a enormi investimenti per aumentare la velocità dei canali dati, arrivando allo standard detto comunemente 3G - sigla che significa "di terza generazione" - che permette velocità superiori

a 1 Mbps e che è ormai diffuso ovunque.

Lo standard successivo, chiamato ovviamente 4G, prevede accessi da mezzi in movimento veloce sino a 100 Mbps e da host in movimento lento o fermi sino a 1 Gbps.

Per ottenere queste prestazioni, il classico canale vocale *a commutazione di circuito* (descritto in un prossimo paragrafo) è stato eliminato e tutto passa attraverso i protocolli Internet.

Con l'aumento esponenziale della velocità dei sistemi radio, le reti di accesso cablate potrebbero diventare velocemente obsolete, se non fosse che la copertura radio del territorio è molto più costosa di quella tradizionale via cavo, sia per i costi di impianto, sia per quelli di manutenzione, e quindi le tariffe richieste dagli ISP sono molto più alte

Struttura di una rete cellulare • Una rete cellulare prevede delle **stazioni base**, collegate a punti di accesso - tipicamente antenne poste su tralicci - che formano celle di copertura, all'interno delle quali si connettono tramite antenne i dispositivi mobili.

Ogni stazione base è connessa tramite un nodo di commutazione alla propria rete dorsale, che utilizza protocolli specifici per le trasmissioni radio. All'interno della rete dorsale si trovano altri particolari nodi, detti **gateway**, il cui scopo è l'interconnessione con reti di altro tipo, in particolare con il sistema telefonico da una parte e con l'Internet dall'altra.

Wimax • Un altro tipo di connessione radio, nato per portare l'Internet a larga banda dove i mezzi fisici non arrivano o hanno prestazioni molto ridotte, è il **Wimax** (*Worldwide Interoperability for Microwave Access*), che permette la connessione punto-multipunto tra stazioni base e host fissi o in lento movimento. A differenza della rete cellulare, infatti, il Wimax non dispone di sistemi di commutazione delle celle, quindi non è in grado di gestire un host che si sposta da una cella all'altra velocemente. Questo limite è stato recentemente eliminato da un emendamento dello standard, che consente di avere host mobili sino a 120 km/h. Al momento, tuttavia, nessuna offerta Wimax pubblica prevede l'utilizzo di questo standard.

Rispetto al 3G e al 4G, il Wimax presenta costi molto più ridotti. Inoltre permette connessioni tra l'utente finale e la stazione base su distanze che possono raggiungere i 50 km e, se si accetta una drastica riduzione delle prestazioni, tra stazioni base e utenti finali che non sono neppure a portata ottica tra loro.

La struttura della rete Wimax è fisicamente simile a quella cellulare, ma con importanti differenze logiche: gruppi di stazioni base sono collegati a nodi interconnessi tra loro e con le altre reti. Poiché, però, i protocolli utilizzati del sistema sono gli stessi dell'Internet, non sono necessari gateway specifici.

I nodi sono chiamati **ASN gateway** (*Access Service Network*), in quanto provvedono alla gestione dei servizi di rete ed alla connessione con i server di autenticazione, che identificano l'utente finale che richiede la connessione.

WiFi • Da ultimo, menzioniamo un particolare tipo di WLAN, commercialmente chiamato **WiFi**, che sta soppiantando la rete cablata nelle abitazioni e nelle piccole strutture, e ciò grazie alle prestazioni sempre più elevate ed alla grande facilità di installazione e di configurazione: lo standard WiFi verrà approfondito nel capitolo 6.

3 Network Core - commutazione di pacchetto e di circuito

Commutazione di pacchetto • Dopo aver brevemente descritto le reti di accesso, dedichiamoci al nucleo della rete, ossia quella trama di *packet switch*, generalmente router, e di *communication link* che permette la comunicazione tra qualunque host connesso all'Internet.

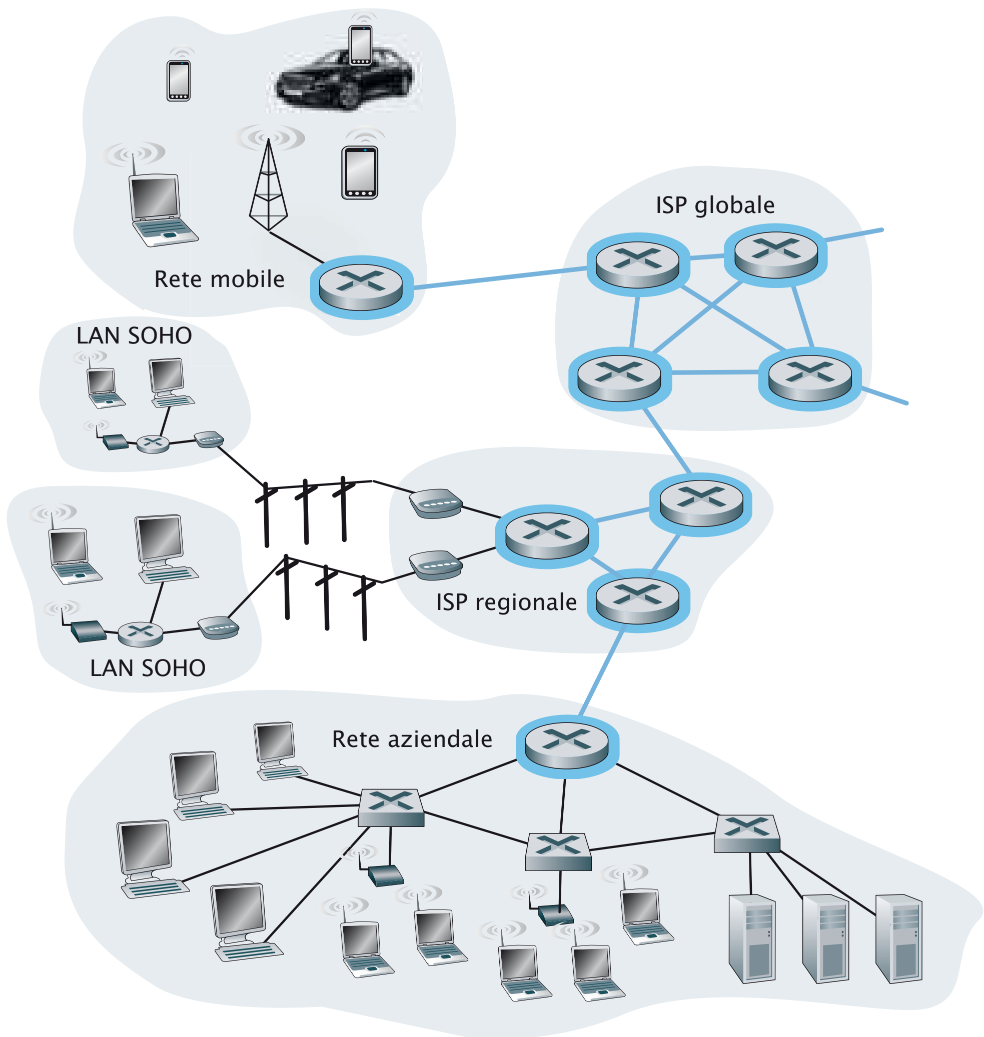


Figura **6**
Network Core.

Caratteristica fondamentale di questa struttura è che i pacchetti vengono inviati da un nodo all'altro sempre alla massima velocità permessa del collegamento fisico. In questo modo, la banda disponibile è sempre completamente sfruttata e la velocità di connessione tra i due host dipende, in teoria, solo dalla larghezza di banda della connessione più lenta e dal numero di host che stanno accedendo contemporaneamente alla connessione.

Quindi, se uno switch sta inviando un pacchetto di *L* bit su un link con velocità di *R* bit/secondo, il tempo di trasmissione sarà *L*/*R* secondi.

Store and Forward • La maggior parte degli switch utilizza la modalità di trasmissione detta ***Store and Forward*** (letteralmente *accumula e inoltra*): in questa modalità, lo switch deve ricevere l'intero pacchetto ad un suo ingresso prima di iniziare a trasmettere il primo bit del pacchetto stesso su una sua uscita.

Ne deriva che, tra l'inizio della ricezione e l'inizio della ritrasmissione, passa un certo tempo, e di conseguenza si genera un ritardo nella trasmissione dovuto all'accumulo del pacchetto all'interno dello switch.

Facciamo un esempio.

Supponiamo che la ricezione del pacchetto inizi al tempo 0 secondi. Dopo un tempo pari a L/R secondi la ricezione è completata, e l'intero pacchetto si trova accumulato nello switch. Ponendo che non esistano ritardi tra la fine della ricezione e l'inizio della trasmissione, e che

i due link abbiamo la stessa larghezza di banda, a questo punto lo switch comincia a trasmettere alla stessa velocità: il tempo totale di trasmissione del pacchetto dall'host sorgente all'host destinatario, quindi, risulta pari a $2L/R$ secondi.

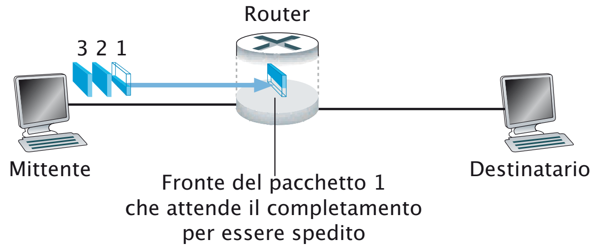


Figura **7**
Store and Forward.

Se i pacchetti da trasferire sono 3, vediamo che al tempo $2L/R$ lo switch ha inviato il primo pacchetto e ricevuto il secondo; al tempo $3L/R$ ha inviato il secondo pacchetto e ricevuto il terzo; al tempo $4L/R$ ha trasmesso anche il terzo pacchetto. Ne deriva, quindi, che il tempo di transito d_e di un pacchetto che passa per N link è:

$$d_e = N\,L/R$$

Il termine d_e sta per ***delay end-to-end***, ossia *ritardo punto-punto*.

Osserviamo anche che se lo switch incominciasse a ritrasmettere un pacchetto appena ricevuto il primo bit del successivo, il ritardo si ridurrebbe a $(N\text{-}1) \cdot L/R$.

Ciò però non può realizzarsi, in quanto il processo di routing effettuato dallo switch necessita di una elaborazione del pacchetto prima della sua ritrasmissione, come vedremo più avanti.

Ritardi di accodamento e perdite di pacchetti • Le cose, in realtà, non sono affatto così semplici. Ogni switch ha molti link di ingresso e uscita: per ognuno di essi, è presente un **buffer** di uscita, che accumula i pacchetti che stanno per essere inviati; questi buffer hanno la fondamentale funzione di contenere i pacchetti che devono essere trasmessi sul link, ma che trovano l'uscita occupata da altri pacchetti di provenienza diversa, ma con la stessa destinazione.

Quindi, oltre ai ritardi dovuti allo *store and forward*, se ne accumulano ulteriori, dovuti al tempo di accodamento nel buffer. Questi ritardi sono variabili e dipendono dal livello di congestione della rete, oltre che da eventuali differenze di larghezza di banda tra il link di ingresso e il link di uscita.

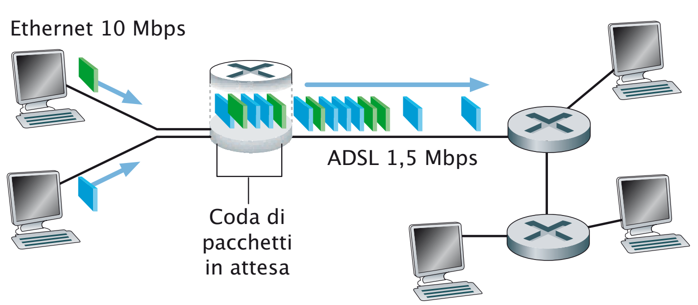


Figura **8**
Ritardo di accodamento.

Poiché lo spazio nel buffer è finito, può accadere che un pacchetto in arrivo trovi il buffer completamente occupato. In questo caso, vi sarà una perdita di pacchetto: a seconda della struttura dello switch, potrà essere gettato via quello appena arrivato o eliminato uno già presente nel buffer.

Esamineremo più dettagliatamente queste problematiche nelle pagine successive.

Commutazione di circuito • Sino a qui abbiamo descritto le basi di funzionamento di un sistema di comunicazione a **commutazione di pacchetto**, in cui l'informazione viaggia suddivisa in pacchetti che vengono singolarmente instradati e ricomposti nel punto di arrivo.

Esiste, però, un altro tipo di sistema di comunicazione, chiamato a **commutazione di circuito**.

Nelle *reti a commutazione di circuito*, le risorse necessarie lungo il percorso per instaurare la comunicazione tra due host, cioè buffer e banda disponibile, sono riservate per tutta la durata della sessione di comunicazione, quindi non devono essere condivise con altre sessioni tra host diversi.

Nelle *reti a commutazione di pacchetto*, invece, le risorse non sono riservate: in questo modo, ogni link viene sempre utilizzato al massimo delle possibilità e una connessione non toglie alcuna disponibilità di spazio alle altre. In compenso, la velocità di trasmissione è a priori ignota, a causa di tutti i ritardi che abbiamo appena descritto.

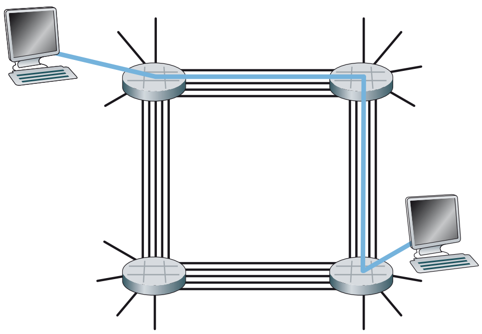


Figura **9**
Commutazione di circuito.

Un tipico esempio di rete a commutazione di circuito è il sistema telefonico tradizionale. Quando un utente ne chiama un altro, prima di poter parlare la rete dovrà stabilire una connessione tra il chiamante ed il chiamato e tutti gli apparati di commutazione coinvolti manterranno la connessione tra i due sino al termine della comunicazione.

In telefonia, questa connessione viene chiamata **circuito**, da cui il termine "a commutazione di circuito". Quando la rete stabilisce il circuito, riserva anche una banda costante attraverso tutti i link per la durata della connessione: per questa ragione, i dati possono essere trasferiti a una velocità costante garantita, senza necessità di buffer e senza attese di disponibilità.

Multiplexing • Ovviamente, se ogni circuito di comunicazione occupasse tutta la banda disponibile sui link coinvolti, l'intero sistema andrebbe in congestione in brevissimo tempo: per ovviare a questo problema, ogni link può ospitare un certo numero di circuiti contemporaneamente, utilizzando la tecnica del **multiplexing**.

Esistono due tipi di multiplexing: quello *a divisione di frequenza* o **FDM** e quello *a divisione di tempo* o **TDM**.

FDM • Come dice il nome, nel multiplexing FDM la banda di frequenze supportate dal link viene suddivisa in bande di ampiezza minore, ciascuna dedicata a un circuito: nelle reti telefoniche, la banda dedicata ad ogni comunicazione ha, tipicamente, un'ampiezza di 4 kHz. Generando **portanti** a frequenze multiple della larghezza di banda, è possibile instaurare contemporaneamente un certo numero di connessioni. Un tempo, tutte le reti telefoniche utilizzavano l'FDM, mentre oggi il segnale audio viene digitalizzato e trattato con tecnica TDM.

Il multiplexing a divisione di frequenza, aggiornato con tecnologie che permettono larghissime bande passanti ai link e la modulazione digitale delle portanti, è utilizzato per l'ultimo miglio delle connessioni DSL, per le reti cellulari 4G e per la TV digitale.

TDM • In un link TDM, il tempo è diviso in **frame** di durata fissa, ognuno dei quali è a sua volta diviso in un numero fisso di parti, chiamate **time slot**.

Quando la rete stabilisce una connessione attraverso un link, dedica ad essa - e solo ad essa - uno specifico time slot all'interno di ogni frame, sino alla fine della connessione.

Maggiore è il numero di time slot in cui è suddiviso il frame, più alta è la quantità di circuiti che può essere veicolata dal link.

In pratica, utilizzando l'FDM ogni circuito occupa in continuazione una piccola parte della banda disponibile, mentre utilizzando il TDM ogni circuito occupa tutta la banda disponibile ma per piccoli intervalli di tempo tra loro equidistanti.

Da quanto detto appare chiaro come la commutazione di pacchetto, quando applicabile al tipo di comunicazione che si vuole stabilire, sia molto più efficiente della commutazione di circuito, in quanto nella prima non esistono tempi morti nelle connessioni, mentre nella seconda l'occupazione dovuta al circuito permane anche in assenza di passaggio di dati.

Se guardiamo alla certezza della comunicazione, tuttavia, ci rendiamo conto facilmente che la commutazione di circuito garantisce una larghezza predefinita e costante, mentre la commutazione di pacchetto, mirando a ottenere la massima velocità, non può tuttavia intrinsecamente garantire il risultato effettivo, a causa delle possibili congestioni dei link.

4 Ritardo, perdita e throughput

Una rete ideale dovrebbe fornire i propri servizi in modo istantaneo, trasportando quantità illimitate di dati tra qualunque host, senza ritardi e senza perdite: nel mondo reale, ovviamente, ciò non corrisponde al vero. Abbiamo già visto come la struttura dei router ed il loro funzionamento introducano forzatamente ritardi e possano causare perdite di dati.

Vedremo ora una panoramica di quello che accade ai pacchetti lungo il loro percorso nella rete.

Mentre un pacchetto viaggia dall'host sorgente all'host destinatario attraverso una serie di nodi, si trova a subire diversi tipi di ritardo in ogni nodo lungo il percorso: i più importanti sono il ***ritardo di processo***, il ***ritardo di accodamento***, il ***ritardo di trasmissione*** ed il ***ritardo di propagazione***. Il loro complesso va a costituire il ***ritardo nodale totale***.

Poiché le prestazioni di molte applicazioni di rete dipendono da questi ritardi, è necessario comprenderne la natura e l'importanza.

Ritardo di processo • Quando un pacchetto arriva all'ingresso di un router, questo

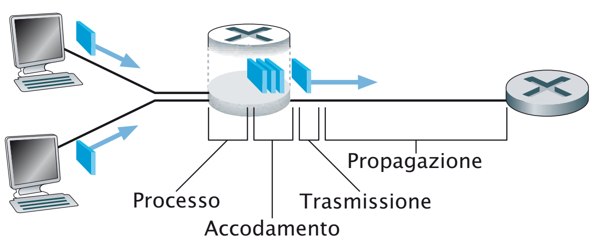


Figura **10**
Ritardo nodale totale.

ne esamina l'header per determinare a quale uscita inviarlo. Per fare ciò, necessita di un certo tempo, che costituisce la parte sostanziale del ritardo di processo. A questo ritardo, poi, si può aggiungere quello dovuto ad altre operazioni che il router compie sul pacchetto, ad esempio il controllo a livello dei singoli bit degli errori, che possono essersi generati durante la trasmissione dal nodo precedente.

Nei router attuali, i ritardi di processo sono dell'ordine dei microsecondi.

Ritardo di accodamento • Dopo aver processato il pacchetto, il router lo dirige al buffer corrispondente all'uscita scelta: il ritardo di accodamento è il tempo per il quale il pacchetto attende nel buffer prima di essere trasmesso.

Se il buffer è vuoto e nessun altro pacchetto è in fase di trasmissione sul link, allora il ritardo è pari a zero.

Se, invece, il buffer è affollato e vi è molto traffico sul link, allora il ritardo sarà proporzionalmente più lungo. Poiché i pacchetti sono trasmessi dal router in modalità **FIFO** (*First In First Out*), ossia sono ritrasmessi nella stessa sequenza con cui sono stati ricevuti, il nostro pacchetto dovrà attendere che tutti i pacchetti che lo precedono siano trasmessi.

I ritardi di accodamento possono variare, in pratica, da qualche microsecondo a qualche millisecondo.

Ritardo di trasmissione • Quando il pacchetto è arrivato in cima al buffer, viene immesso sul link, sequenziando i bit.

La velocità con cui avviene la trasmissione dipende dalla larghezza di banda del link, che comprende quella delle porte di uscita del router e quelle di ingresso nel router successivo: come abbiamo già visto, se il pacchetto è composto di L bit e la velocità di trasmissione è R bit/secondo, il tempo di trasmissione risulta pari a L/R secondi. Anche per il ritardo di trasmissione, il valore può variare da qualche microsecondo a qualche millisecondo.

Ritardo di propagazione • Una volta spinto nel link, ogni bit del pacchetto impiega un certo tempo a percorrere la distanza che lo separa dalla porta di ingresso del router successivo.

Questo tempo dipende dalla velocità di propagazione del segnale all'interno del mezzo trasmissivo: poiché si tratta di campi elettromagnetici, questa velocità è uguale o poco inferiore a c (la velocità della luce), che è pari a circa 300.000 km/s.

Il ritardo di propagazione, quindi, è pari a d/v, dove d è la lunghezza del link e v la velocità di propagazione. Nelle reti geografiche, con nodi molto distanti, i ritardi di propagazione possono arrivare a qualche millisecondo.

È importante capire la differenza fondamentale tra ritardo di trasmissione e ritardo di propagazione. Il primo è il tempo necessario al router per spingere sul link l'intero pacchetto, e dipende sia dalla dimensione in bit del pacchetto, sia dalla larghezza di banda del link e delle porte di trasmissione e ricezione ai suoi estremi. Il secondo è il tempo che un bit impiega a percorrere il link, e non ha nulla a che vedere con i parametri appena esposti ma dipende soltanto dalla lunghezza del link e dalla velocità di propagazione delle onde elettromagnetiche nel mezzo fisico che costituisce il link (rame, fibra ottica oppure aria).

Come abbiamo già detto, il ritardo nodale totale è la somma di tutti i ritardi esaminati: se indichiamo con d_{proc}, d_{queue}, d_{trans}, d_{prop} rispettivamente i ritardi di processo, accodamento, trasmissione e propagazione, il ritardo totale d_{nodal} sarà dato da:

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Il contributo dei vari ritardi sul valore totale varia significativamente. In una LAN, ad esempio, d_{prop} è insignificante, dato che i link sono lunghi al massimo un centinaio di metri e, quindi, sono percorsi in qualche nanosecondo; al contrario, in una connessione satellitare con satelliti geostazionari in orbita a 36.000 km, il ritardo di propagazione d_{prop} supera il decimo di secondo.

Sempre in una LAN, il ritardo di trasmissione d_{trans} è trascurabile, dato che al giorno d'oggi velocità di 1 Gbps sono la regola anche per il link terminali. Invece d_{trans} diventa fondamentale quando vengono trasmessi grandi pacchetti su linee ADSL poco performanti.

Intensità di traffico • Il ritardo di accodamento d_{queue}, spesso detto semplicemente **latenza**, è il componente più importante del ritardo totale, ed è anche quello su cui si è focalizzata l'attenzione dei progettisti, in quanto dipende dall'architettura interna del router.

Vediamo, innanzitutto, che il ritardo di accodamento può variare da pacchetto a pacchetto. Se, ad esempio, dieci pacchetti entrano insieme nel buffer, il primo ad essere trasmesso non soffre di alcuna latenza, mentre l'ultimo, dovendo attendere la trasmissione dei primi nove,

avrà un ritardo relativamente lungo. Per questa ragione, il ritardo di accodamento viene definito in termini statistici: *ritardo medio*, *varianza del ritardo*, *probabilità che il ritardo superi un valore predeterminato*.

L'entità del ritardo di accodamento dipende da molti fattori: il ritmo a cui il traffico giunge al buffer, la velocità di trasmissione del link in uscita e la natura del traffico entrante, ossia se perviene come flusso continuo o se arriva a impulsi.

Poniamo che R sia la velocità di trasmissione (in bit/s), a sia il ritmo di arrivo dei pacchetti, espresso in pacchetti/s e, per semplicità, supponiamo anche che tutti i pacchetti siano composti da L bit: ne deriva che il flusso medio di pacchetti in entrata è $\boldsymbol{La}$.

Sempre per semplicità, poniamo anche il buffer sia infinito, quindi possa contenere qualsiasi numero di pacchetti: otteniamo un rapporto adimensionale $\boldsymbol{La/R}$, che definisce l'occupazione del buffer e, quindi, fornisce un'informazione che permette il calcolo della latenza dei pacchetti.

Questo rapporto viene detto ***intensità di traffico*** e gioca un ruolo molto importante nella stima del ritardo di accodamento.

Consideriamo il caso in cui $\boldsymbol{La/R} > 1$: il ritmo di arrivo del traffico nel buffer è superiore al ritmo di trasmissione, e i pacchetti si accumuleranno in continuazione, portando all'infinito il tempo di latenza.

Consideriamo il caso in cui $\boldsymbol{La/R} \leq 1$: qui non esiste più un accumulo continuo e diventa importante la natura del traffico entrante.

Se i pacchetti arrivano periodicamente, ogni L/R secondi, allora il buffer è sempre libero e non ci sono ritardi; se, invece, i pacchetti arrivano a impulsi, allora può esserci una significativa latenza media.

Supponiamo, ad esempio, che N pacchetti arrivino simultaneamente ogni $(\boldsymbol{L/R})\boldsymbol{N}$ secondi. Come abbiamo visto, il primo pacchetto non ha alcun ritardo, il secondo ha un ritardo L/R e, in generale, l'ennesimo pacchetto ha un ritardo $(n\text{-}1)L/R$ secondi, quindi un ritardo che cresce regolarmente.
Il ritardo medio del gruppo di pacchetti risulta, quindi

$$\frac{1}{N}\sum_{n=1}^{N}\frac{nL}{R}$$

Questi esempi sono accademici e non corrispondono affatto alla realtà: tipicamente il processo di entrata del traffico non è periodico, ma casuale: l'arrivo dei pacchetti non segue alcuno schema, e i pacchetti sono separati da tempi non prevedibili.

Per questa ragione, il rapporto La/R non è sufficiente per definire compiutamente il ritardo di accodamento. Tuttavia, l'intensità di traffico può essere usata per predire il comportamento generale di una rete: quando La/R è prossimo allo zero, anche la latenza è nulla o quasi, mentre quando si avvicina a 1, il comportamento della rete peggiora costantemente. Infatti, anche se l'intensità di traffico fosse mediamente inferiore a 1, vi saranno dei momenti in cui essa supererà l'unità nei quali, quindi, la coda si allungherà. Anche se, nei momenti di minor traffico, la coda si riduce, nel tempo il ritardo continuerà a crescere con andamento asintotico sino a giungere a momenti di congestione della rete: per evitare o risolvere tale eventualità, esistono strategie che descriveremo nei prossimi capitoli.

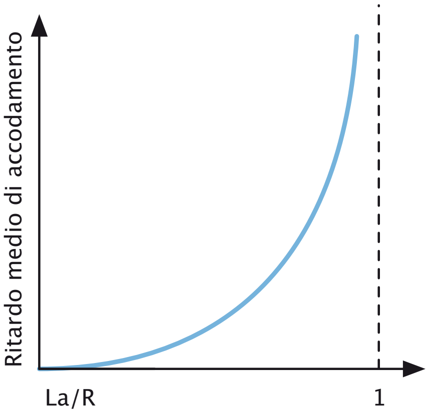


Figura **11**
Rapporto tra ritardo di accodamento e intensità di traffico.

Perdita di pacchetti • Sin qui, per semplificare il ragionamento abbiamo supposto che il buffer del router sia infinito, e quindi sia in grado di accogliere tutti i pacchetti entranti, indipendentemente dal loro numero. In realtà, benché dipenda grandemente dall'architettura e dal costo del router, la dimensione del buffer è comunque finita. Per questa ragione, in effetti, la latenza dei pacchetti non tende all'infinito quando l'intensità di traffico tende a 1: infatti, quando il buffer è pieno, il pacchetto entrante non trova più posto e il router lo elimina, quindi il pacchetto viene perso.

Dal punto di vista degli host, la perdita di un pacchetto può essere vista come un pacchetto immesso nella rete, ma mai riemerso a destinazione. Poiché la percentuale di pacchetti persi aumenta all'aumentare dell'intensità di traffico, le prestazioni di un nodo sono misurate anche in termini di "*probabilità di perdita*" di pacchetti.

Throughput • Si tratta di un termine intraducibile in Italiano - la cui traduzione letterale sarebbe, in questo caso, *passaggio attraverso* - che indica la quantità di bit ricevuta con successo alla fine di un canale di comunicazione. Equivale alla portata di una conduttura di tipo idraulico.

Immaginiamo di inviare dall'host A all'host B un grosso file, ad esempio un video: il throughput istantaneo è la velocità, espressa in bit/s, a cui in ogni istante l'host B sta ricevendo il file; il throughput medio, invece, è dato dalla dimensione del file, divisa per il tempo totale di ricezione.

Se consideriamo, all'interno di una rete, un percorso tra l'host A e l'host B composto da diversi nodi e diversi link, il throughput della connessione non può, ovviamente, essere superiore alla capacità di trasporto del link più lento, che viene perciò detto *collo di bottiglia*.

Facciamo un esempio.

Per semplicità, immaginiamo un percorso formato solo dai due host e da un unico nodo intermedio e che l'host A sia un server che sta inviando all'host B un file video di 4 MB - ossia di 32 Mb - e che abbia una connessione al router, con larghezza di banda Rs pari a 2Mb/s; a sua volta, il router è connesso all'host B con un link di accesso, avente una banda *R*c limitata a 1 Mbps.

Ammesso che non vi siano ritardi da congestione nel nodo, il tempo di trasferimento è

$$F/min\{Rs, Rc\}$$

ossia 32 Mb/1 Mbps = 32 secondi. Il throughput della tratta è, quindi, pari a 1Mbps.

Nella pratica, però, il throughput non è determinato solo dal link più lento della connessione, ma anche dal traffico concomitante: in particolare, anche un link con grande larghezza di banda può diventare comunque un collo di bottiglia se molti altri flussi stanno passando contemporaneamente attraverso di esso.

5 Protocolli e modelli di servizio

Le reti possono essere strutture molto complesse, e l'Internet non fa eccezione: per organizzarle, controllarle e farle funzionare è necessario un approccio che in qualche modo ne semplifichi la rappresentazione e permetta di intervenire in modo mirato sulle varie componenti: la soluzione adottata dagli esperti è stata quella di adottare un modello di struttura a più livelli. Ne abbiamo parlato, anche se sommariamente, nei primi anni di questo corso.

Un esempio • Per capire meglio cosa significa generare un modello a livelli di un sistema, usiamo come esempio pratico le linee aeree: quale modello possiamo utilizzare per descrivere il complesso insieme che comprende figure molto diverse e apparentemente non in contatto tra loro, quali agenti di biglietteria, personale aeroportuale, piloti, controllo del traffico, aeroplani e rotte aeree?

Possiamo partire dalle azioni che compiamo per volare da un punto all'altro: compriamo i biglietti, facciamo il check-in, andiamo al gate, saliamo sull'aereo; l'aereo decolla, segue una

rotta e atterra; dopo l'atterraggio scendiamo dall'aereo, recuperiamo i bagagli, andiamo al gate e, se il viaggio non ci ha soddisfatto, ci lamentiamo con chi ci ha venduto i biglietti.

Se proviamo a formalizzare quanto descritto in una struttura a livelli, otteniamo un modello molto chiaro, che mette in correlazione le azioni compiute all'inizio e alla fine del viaggio suddividendole per servizi compiuti verso il viaggiatore.



Figura **12**
Modello a livelli del traffico aereo.

Se consideriamo i singoli servizi, vediamo come non importi la modalità con cui sono erogati, purché non cambi il servizio in quanto tale: in questo modo, è possibile intervenire su ogni singolo livello senza propagazione sui livelli adiacenti. Questa strategia consente di poter intervenire con miglioramenti e potenziamenti del livello in questione senza che si presentino ripercussioni, e quindi necessità di interventi, sui livelli adiacenti.

Stratificazione dei protocolli • Nelle reti, i protocolli di comunicazione sono stratificati su più livelli, così come l'hardware e il software che li realizzano: ogni protocollo appartiene ad un dato livello, come le funzioni che abbiamo visto nella linea aerea. Ogni livello svolge dei servizi, che offre al livello superiore e, per far ciò, utilizza i servizi di quello inferiore.

Abbiamo accennato negli anni scorsi al modello ISO-OSI, composto di sette livelli, che è il modello di riferimento di tutte le reti.

L'Internet, tuttavia, utilizza un proprio modello semplificato a cinque livelli, che meglio ne rappresenta l'effettivo funzionamento, e che viene sempre più usato anche per sistemi di minori dimensioni. Eccone la descrizione.

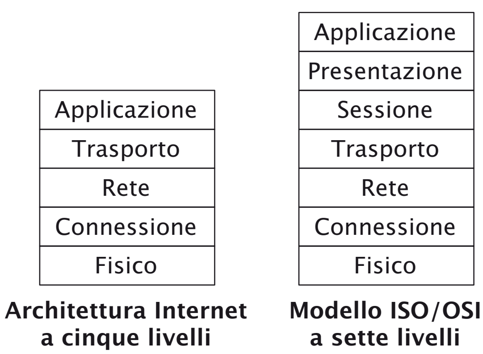


Figura **13**
Raffronto tra architettura Internet e modello ISO/OSI.

Livello applicazione (livello 5) • È il livello in cui risiedono le applicazioni e, quindi, è distribuito sugli host connessi alla rete. Qui vengono gestiti i protocolli di più alto livello, quali HTTP, SMTP, FTP, che provvedono ad effettuare le richieste ed i trasferimenti tra host. Anche i protocolli per la generazione di un'interfaccia comprensibile all'utente risiedono a questo livello.

Un'applicazione residente in un host utilizza i protocolli a questo livello per scambiare blocchi di informazione con un'applicazione residente in un altro host: chiameremo **messaggio** questo blocco di informazione al livello applicativo.

Livello trasporto (livello 4) • Questo secondo livello trasporta i messaggi tra i punti terminali delle applicazioni.

Nell'Internet, il protocollo di trasporto è TCP, che fornisce servizi fondamentali, quali la consegna garantita dei messaggi a destinazione ed il controllo di flusso, ossia fa in modo che la velocità di trasmissione del mittente e quella di ricezione del destinatario si equivalgano.

TCP, inoltre, suddivide i messaggi di grandi dimensioni in blocchi più piccoli, che vengono chiamati **segmenti**, e fornisce un meccanismo di controllo delle congestioni, adeguando la velocità di trasmissione dell'host alle condizioni della rete.

Livello rete (livello 3) • È il livello che prende i segmenti consegnati dal livello superiore e li inserisce in altri blocchi chiamati **datagram**, quindi fornisce il servizio di consegna all'indirizzo richiesto dal livello trasporto.

Nell'Internet, il protocollo che fornisce questi servizi è IP, che definisce i campi all'interno del datagram e come gli host e i router si comportino rispetto ad essi.

Esiste un solo protocollo IP, quindi tutti i componenti dell'Internet che hanno un livello rete devono essere in grado di gestirlo. Ormai, anche le più piccole LAN domestiche lo utilizzano: è necessario per la connessione all'Internet e, anche se questa non ci fosse, non c'è alcuna ragione per utilizzare un protocollo diverso. Il livello rete contiene anche i protocolli di instradamento (*routing*), che determinano i percorsi che i datagram seguono tra mittente e destinatario. Esistono molti protocolli di instradamento e all'interno di ogni rete è possibile sceglierne uno specifico: è il protocollo IP che funge da collante, uniformando i risultati del servizio.

Livello connessione (livello 2) • Il livello rete instrada i datagram attraverso i router che si trovano tra il mittente e il destinatario: per spostare i pacchetti da un nodo al successivo, si appoggia ai servizi forniti dal livello connessione. In particolare, ad ogni nodo il livello rete passa i datagram al livello connessione, che li consegna al nodo successivo e li rimanda al livello superiore.

I servizi forniti dipendono dallo specifico protocollo di connessione utilizzato: ad esempio, alcuni protocolli di livello connessione forniscono la consegna garantita dei datagram fra un nodo e un altro. Ogni link può utilizzare protocolli diversi, quindi lungo il percorso un datagram può incontrare molti protocolli, che forniscono al livello superiore servizi diversi, pur comunicando sempre nello stesso modo: sono protocolli di connessione, fra gli altri, Ethernet, WiFi, PPP, DSL. I pacchetti utilizzati dal livello connessione sono chiamati **frame**.

Livello fisico (livello 1) • Mentre il compito del livello rete è trasferire interi frame da un nodo al nodo adiacente, quello del livello fisico è trasportare i singoli bit che compongono un frame. I protocolli di questo livello discendono da quelli del livello superiore e dipendono dal mezzo fisico di trasmissione. Ad esempio, l'Ethernet ha molti protocolli di livello fisico: per il doppino in rame, per il cavo coassiale, per la fibra ottica, a loro volta suddivisi in funzione delle velocità di trasmissione raggiungibili e nelle distanze percorribili.

Incapsulamento • Come accade negli host, anche gli switch e i router hanno hardware e software organizzato secondo i livelli di rete ma, a differenza di essi non provvedono alla gestione di tutti i livelli: tipicamente, gestiscono solo quelli inferiori. I router, ad esempio, devono gestire il protocollo IP, che è di livello 3, mentre gli switch normalmente si limitano a lavorare a livello 2. In realtà, esistono switch di livello 3 e superiore, ma il loro utilizzo è limitato a situazioni particolari e, comunque, non è indispensabile per il funzionamento dell'Internet.

Tuttavia, anche l'informazione contenuta nei protocolli a più alto livello deve giungere

dal mittente al destinatario, altrimenti le applicazioni non potrebbero rispondere alle richieste: gli host, quindi, devono poter ricostruire i frame dai singoli bit, ricavare i datagram dai frame, estrarre dai datagram i segmenti e, a loro volta, riunirli a formare il messaggio.

Per ottenere ciò si utilizza la tecnica detta **incapsulamento**.

Nell'host mittente, il messaggio **M** generato dall'applicazione a livello 5 è passato al livello sottostante che, nel caso più semplice, lo prende ed aggiunge in testa ulteriori dati - il già citato *header* (**H**) - che saranno utilizzati dai protocolli del livello di trasporto nell'host destinatario, generando così un segmento.

Il segmento, quindi, contiene il messaggio proveniente dal livello superiore: si dice che il messaggio è **incapsulato** nel segmento. La stessa tecnica viene utilizzata nei passaggi ai livelli inferiori.

I protocolli di ogni livello provvedono a inserire header diversi, i quali trasportano informazioni ulteriori, che chiamiamo rispettivamente **Ht**, **Hr**, **Hc** per i livelli di trasporto, rete e connessione.

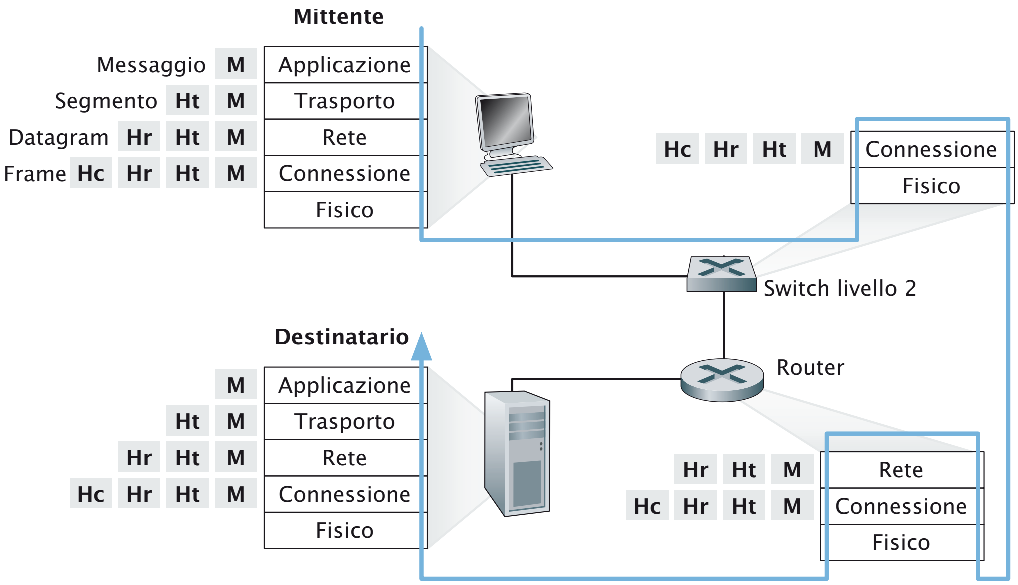


Figura **14**
Incapsulamento ai vari livelli.

Traceroute

Il *Traceroute* è un piccolo programma che può tracciare il percorso tra due host, rilevandone anche i ritardi totali. Il suo funzionamento è molto semplice: quando il mittente specifica l'indirizzo di destinazione, Traceroute invia verso di essa un serie di pacchetti speciali i quali, quando attraversano un nodo, fanno sì che il router rimandi al mittente un messaggio contenente il proprio nome e indirizzo. In questo modo, il mittente può ricostruire il percorso compiuto dai pacchetti lungo la rete ed i ritardi che avvengono nelle tratte tra i singoli router.

Nell'esempio proposto online, potete trovare un output di Traceroute (si tratta del percorso tra i nostri computer e il sito www.zanichelli.it).

Nel percorso tra mittente e destinatario si contano 19 router. In realtà, l'host di destinazione appartiene ai servizi AWS di Amazon, come si vede dall'ultima riga del listato: per questo Traceroute non riesce a tracciare il tratto di percorso finale, che si trova all'interno della LAN di Amazon. Si notino anche le righe 11 e 12, che mostrano il nome dell'ISP che fornisce la connessione (Fastweb).

Il programma mostra tre valori di ritardo per ogni link, in quanto è predisposto per compiere tre tentativi di invio in sequenza per ogni link. Notiamo che alcuni link hanno ritardi praticamente uguali, mentre altri presentano notevoli differenze, dovute evidentemente all'andamento del traffico al loro interno.

A1-01
Un esempio di Traceroute

Concetti essenziali

- La struttura di Internet è descrivibile come un insieme di nodi ospite, communication link e packet switch.
- Un nodo ospite è collegato ad un altro nodo ospite attraverso un percorso sul quale si trova un numero imprecisato di packet switch.
- Ogni packet switch riceve e ritrasmette pacchetti reinstradandoli sulla base delle informazioni contenute negli header.
- La rete di accesso connette fisicamente un host al primo router dell'Internet, detto edge router.
- Esistono connessioni via cavo, via fibra ottica, via radio.
- Le comunicazioni possono avvenire secondo due tecniche: commutazione di pacchetto e commutazione di circuito.
- La commutazione di pacchetto sfrutta sempre completamente la banda disponibile, decompone il messaggio in pacchetti e li ricompone all'arrivo.
- La commutazione di circuito riserva un canale fisso tra due punti dopo aver instaurato il link.
- La somma delle componenti che possono causare ritardi nella trasmissione in un nodo prende il nome di ritardo nodale totale.
- Il ritardo di processo è causato dal tempo impiegato dal nodo affinché l'header venga esaminato e si determini a quale uscita inviarlo.
- Il ritardo di accodamento è il tempo di attesa del pacchetto sul buffer in uscita, in attesa di essere trasmesso.
- Il ritardo di trasmissione è legato alla effettiva velocità di trasmissione, funzione della larghezza di banda del link.
- Il ritardo di propagazione è funzione della velocità del segnale nel mezzo trasmissivo prescelto.
- Il throughput di una trasmissione è la quantità di bit ricevuta con successo in rapporto al tempo di trasmissione.
- Internet utilizza un modello semplificato a cinque livelli.

Test

1 Dire se le seguenti affermazioni sono vere o false.
L'Internet può essere descritta come:
- A uno speciale sistema operativo per i sistemi mobili V F
- B un'infrastruttura che fornisce servizi ad applicazioni distribuite V F
- C un'applicazione che richiede servizi alle infrastrutture V F
- D un'enciclopedia in cui tutto il sapere è reso accessibile a tutti V F

2 Dire se le seguenti affermazioni sono vere o false.
Un protocollo di rete definisce:
- A il tipo di linguaggio con cui deve essere scritta l'applicazione V F
- B la massima distanza geografica tra due nodi V F
- C la sequenza di trasmissione dei files tra due PC V F
- D il formato e l'ordine dei messaggi scambiati V F

3 Dire se le seguenti affermazioni sono vere o false.
Gli apparati collegati via Internet, quali ad esempio i PC, gli smartphone ed i tablet, sono detti:
- A host system V F
- B most system V F
- C start system V F
- D star system V F

4 Dire se le seguenti affermazioni sono vere o false.
Sul percorso tra due nodi si possono trovare:
- A postal switch, in numero noto e prefissato V F
- B switching packs, di tipo analogico V F
- C packet switches, in numero imprecisato V F
- D soft packs, in numero variabile in funzione della fascia oraria V F

5 Dire se le seguenti affermazioni sono vere o false.
Il primo router Internet cui un host si connette prende il nome di:
- A head router V F

B route 66 V F
C routing switch V F
D edge router V F

6 **Dire se le seguenti affermazioni sono vere o false.**
DSL significa:
A Digital Status Link V F
B Double Sync Lan V F
C Digital Subscriber Line V F
D Dropping Star Level V F

7 **Dire se le seguenti affermazioni sono vere o false.**
Nella sigla ADSL, la lettera A significa:
A asimmetrico V F
B asintotico V F
C algoritmico V F
D architettonico V F

8 **Dire se le seguenti affermazioni sono vere o false.**
In una rete cellulare troviamo:
A stazioni base, punti di accesso, dorsali V F
B concetti base, punti di accesso, spine V F
C cablaggi in fibra di tipo punto-punto V F
D centrali telefoniche, antenne, ricevitori TV V F

9 **Dire se le seguenti affermazioni sono vere o false.**
In una rete a commutazione di pacchetto:
A ogni messaggio viene decomposto in segmenti e ne vengono trasmessi solo quelli di lunghezza inferiore a 100 byte V F
B l'informazione è suddivisa in pacchetti che vengono instradati singolarmente V F
C la stazione trasmittente deve sempre crittografare ogni singolo pacchetto V F
D l'informazione è contenuta solo all'inizio della serie di pacchetti, nell'header V F

10 **Dire se le seguenti affermazioni sono vere o false.**
FDM e TDM sono:
A due diverse frequenze di trasmissione V F
B due tipi di packet switch V F
C due tipi di multiplexing V F
D due diverse reti wireless V F

11 **Dire se le seguenti affermazioni sono vere o false.**
Una rete ideale dovrebbe operare:
A senza alcun tipo di protocollo V F
B solo in modalità wireless V F
C meglio nei giorni festivi V F
D senza ritardi e senza perdite V F

12 **Dire se le seguenti affermazioni sono vere o false.**
Il tempo che un pacchetto impiega a percorrere il tratto tra l'uscita di un router e l'ingresso del successivo prende il nome di:
A ritardo di concentrazione V F
B corruption delay V F
C ritardo di accesso V F
D ritardo di propagazione V F

13 **Dire se le seguenti affermazioni sono vere o false.**
In una rete:
A può succedere che un pacchetto si perda V F
B i primi 100 pacchetti trasmessi vengono sempre persi V F
C li pacchetti dispari sono la copia di quelli pari, per avere un backup in caso di perdita V F
D non esiste nessuna probabilità di perdita di pacchetti V F

14 **Dire se le seguenti affermazioni sono vere o false.**
Il modello teorico dell'Internet prevede:
A 4 livelli V F
B 5 livelli V F
C 6 livelli V F
D 7 livelli V F

15 **Dire se le seguenti affermazioni sono vere o false.**
La tecnica che costruisce un segmento contenente il messaggio del livello superiore e aggiunge dati per rendere possibile la ricostruzione del messaggio originario prende il nome di:
A packaging V F
B packet wrapping V F
C imbottigliamento V F
D incapsulamento V F

2 Il livello applicazione

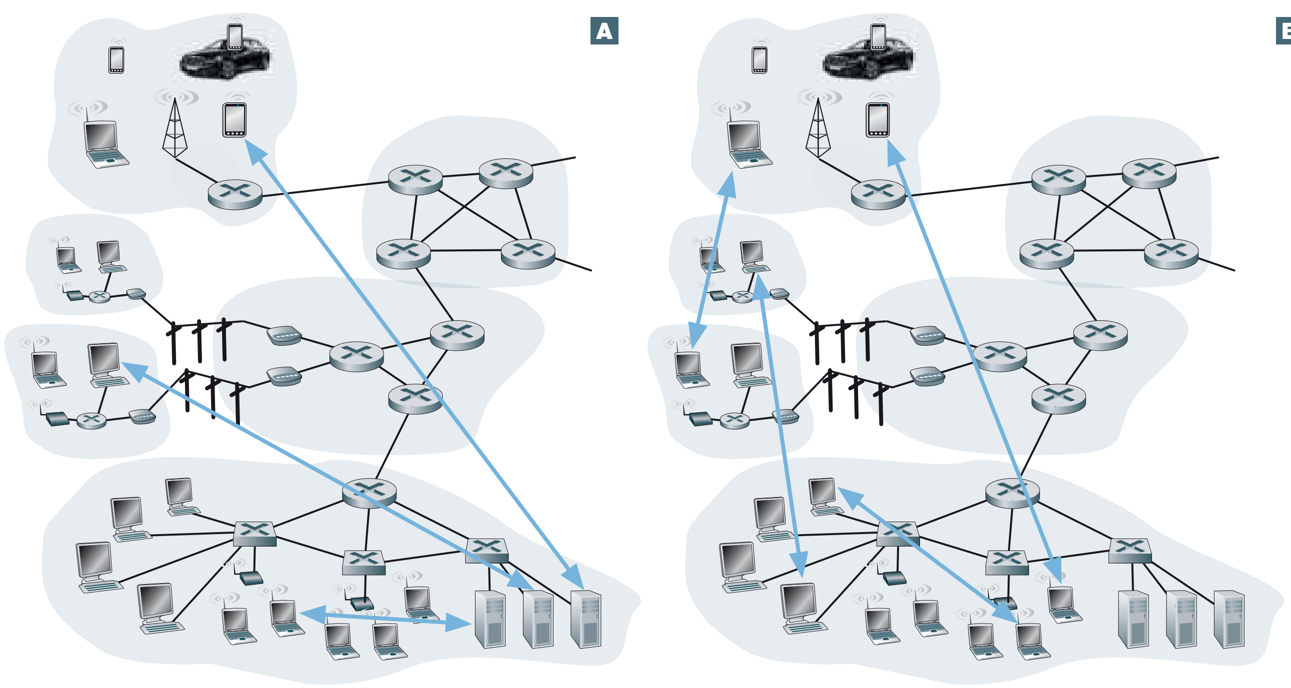
I concetti fondamentali su cui si basa la gestione delle reti, introdotti nei precedenti volumi, vengono qui rivisitati in maniera più approfondita. Il capitolo illustra, evidenziandone punti di forza e punti di debolezza, tecniche diverse che permettono di ottenere risultati analoghi, sia che si tratti di scegliere un'architettura client-server o peer-to-peer, sia che si tratti di scegliere i protocolli di livello applicazione. Grazie a un'analisi tecnica di dettaglio, vengono esplorate le caratteristiche intrinseche che ne rendono alcune migliori, o più efficaci di altre, magari in dati contesti o con determinate condizioni operative al contorno.

1 Architettura e comunicazione tra processi

L'architettura stratificata delle reti permette di sviluppare le applicazioni residenti sugli host senza doversi in alcun modo preoccupare della gestione delle reti stesse: dal punto di vista di un'applicazione, l'architettura di rete è fissa e fornisce uno specifico gruppo di servizi. L'architettura dell'applicazione, invece, è scelta dallo sviluppatore e definisce come l'applicazione stessa è strutturata sui differenti tipi di host. In particolare, la prima scelta da effettuare da parte di chi sviluppa applicazioni in rete è il paradigma secondo il quale gli host comunicano tra loro: l'architettura *client-server* o quella *peer-to-peer*.

Client-server • In questo paradigma esiste un host sempre attivo, detto *server*, che riceve richieste di servizio da altri host, detti *client*. L'applicazione **World Wide Web** ne è un tipico

Figura **1**
[A] Architettura client-server.
[B] Architettura peer-to-peer.



esempio: il browser è l'applicazione client, che invia al Web server le richieste e ne riceve le risposte. Notiamo che in un'architettura di questo tipo, i client non comunicano mai direttamente tra loro. Un'altra caratteristica fondamentale è che il server deve possedere un indirizzo di rete fisso e noto, in modo che ogni client possa contattarlo in qualunque momento.

I server devono essere in grado di soddisfare contemporaneamente le richieste di un numero potenzialmente molto elevato di client. Per questo motivo sono spesso formati da centinaia o migliaia di macchine gestite dai **Service Provider** all'interno di strutture dedicate chiamate **Data center** o **Server Farm**.

Peer-to-peer • In questa architettura, la dipendenza da un server è minima o nulla e l'applicazione utilizza una connessione diretta e nintermittente tra coppie di host, chiamati **peer** (in italiano, letteralmente *pari*). Poiché le coppie di host comunicano senza passare attraverso un server, l'architettura viene chiamata **peer-to-peer** (comunemente indicato come **P2P**). Oltre che per lo scambio diretto di file, il P2P viene utilizzato per sistemi di telecomunicazioni in rete, quali la telefonia su Internet.

Esistono anche applicazioni che utilizzano un'architettura mista, che combina elementi di P2P con strutture client-server: ad esempio, la maggior parte delle applicazioni di messaggistica istantanea utilizzano dei server per tracciare gli indirizzi degli utenti, mentre i messaggi sono passati direttamente tra un host e l'altro.

Comunicazione tra processi • Nel gergo dei sistemi operativi, i programmi che girano su un host sono detti **processi**, e quindi sono i processi che comunicano tra loro: se lo fanno stando all'interno di uno stesso host, utilizzano le regole definite dal sistema operativo; quando, invece, devono comunicare tra loro processi che risiedono su host differenti, devono utilizzare le regole dettate dall'architettura della rete che li connette, regole indipendenti sia dal sistema operativo degli host, sia dalla struttura dei processi stessi.

In una sessione di comunicazione tra due processi, **si dice processo client quello che inizia la comunicazione, ossia quello che contatta l'altro all'inizio della sessione, mentre è chiamato processo server quello che aspetta di essere contattato**.

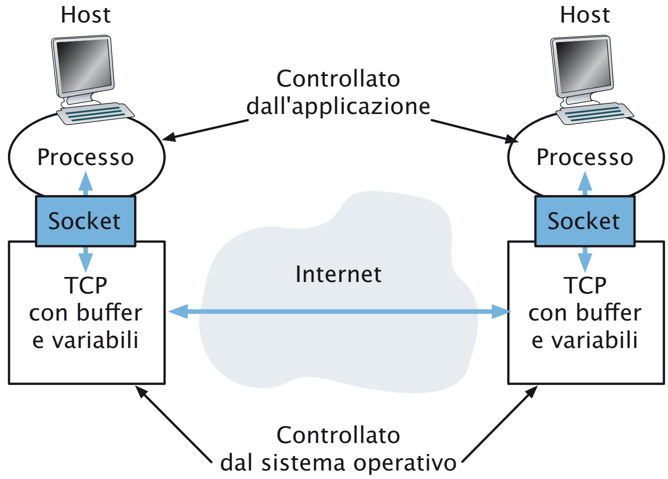
Socket • Per comunicare tra loro, due processi devono utilizzare i servizi messi a disposizione dalla rete sottostante. L'interfaccia software che permette ai processi di inviare e ricevere messaggi sulla rete si chiama **socket** (è il termine con cui, in inglese, si indicano le prese di corrente).

Possiamo definire un socket come il punto in cui un processo accede al canale di comunicazione attraverso una opportuna porta. Dal punto di vista dell'applicazione, il socket è un particolare oggetto su cui vengono scritti i dati da trasmettere e da cui leggere i dati ricevuti.

Chi sviluppa un'applicazione ha il completo controllo di tutto ciò che si trova al livello applicazione del socket, mentre non ne ha su quanto si trova al livello trasporto (al più può decidere il settaggio di qualche parametro): una volta che si è scelto di usare un determinato protocollo di trasporto, l'applicazione non può essere scritta in un modo qualsiasi, ma è obbligata a utilizzare i servizi forniti da quello specifico protocollo.

I protocolli a livello applicazione, quindi, possono essere definiti indipendentemente dalla rete e dai protocolli di trasporto utilizzati, purché si interfaccino correttamente con il socket. Ciò significa che un'applicazione può anche utilizzare protocolli proprietari, accettando il fatto che nessun altro potrà connettersi, oppure utilizzare protocolli standard, permettendo ad altre applicazioni di comunicare in modo trasparente.

Figura **2**
Processi, socket e protocolli di trasporto.



Servizi di trasporto • Le reti - e quindi l'Internet - dispongono di più di un protocollo al livello di trasporto, ma ogni processo al livello superiore può appoggiarsi a uno solo di essi. La scelta viene fatta sulla base delle caratteristiche dei servizi forniti dai diversi protocolli, in particolare:

- l'affidabilità
- il throughput
- il ritardo
- la sicurezza

Affidabilità • Abbiamo visto nel capitolo precedente che, durante il trasferimento in rete, si possono perdere dei pacchetti. Per molte applicazioni, in particolare per quelle che scambiano documenti, la perdita di dati può avere conseguenze anche disastrose: è necessario, quindi, che il protocollo di trasporto utilizzato da queste applicazioni garantisca la consegna integrale e corretta dei dati trasmessi. Se un protocollo offre questo livello di prestazioni, si dice che fornisce un trasferimento dati *affidabile*: in questo caso, il processo che trasmette può limitarsi a passare i dati al socket, avendo la certezza che giungeranno senza errori al processo che riceve.

Se invece il protocollo di trasporto non fornisce un servizio affidabile, alcuni dei dati inviati potrebbero non arrivare mai al ricevente. Esistono applicazioni, però, che sono in grado di sopportare questo inconveniente. Si tratta delle applicazioni dette **loss-tolerant** (letteralmente "*che tollerano le perdite*"), che sono in grado di ricostruire, almeno parzialmente, le stringhe mancanti. Un caso tipico sono le applicazioni multimediali, in cui la perdita si manifesta come un piccolo vuoto che non dà problemi, quando addirittura il dato mancante non viene ricostruito per interpolazione, quindi senza che la perdita di dati determini conseguenze visibili sul messaggio.

Throughput • Nel contesto di una sessione di comunicazione tra due processi attraverso una rete, possiamo definire il *throughput* disponibile come il ritmo al quale il processo mittente può consegnare dati al processo ricevente. Poiché normalmente le risorse di rete sono condivise da più sessioni che iniziano e terminano in modo casuale, il throughput disponibile fluttua nel tempo. Un protocollo di trasmissione può fornire a un processo che lo richieda un ritmo di trasferimento minimo, ossia un throughput disponibile garantito: con questo servizio, l'applicazione può richiedere un throughput garantito di r bit/s, e il protocollo di trasmissione garantirà un throughput disponibile di almeno r bit/s.
Le applicazioni che necessitano di un throughput minimo garantito sono dette **bandwidth-sensitive** (cioè *sensibili alla banda*). Non è sorprendente il fatto che le applicazioni multimediali, molto tolleranti sull'affidabilità del trasferimento, non lo siano sulla disponibilità di banda, mentre le applicazioni critiche sull'affidabilità non abbiano alcuna necessità di throughput garantito, adattando il trasferimento alle disponibilità della rete, arrivando anche al provvisorio azzeramento del flusso dei dati.

Ritardo • Un protocollo di trasporto può anche fornire garanzie sul ritardo con cui consegna i dati attraverso la rete. Questo tipo di servizio è essenziale per le applicazioni interattive in tempo reale, quali telefonia, videoconferenza, ambienti virtuali, che richiedono specifiche stringenti nella consegna dei dati per poter funzionare correttamente. In questi casi, il protocollo può garantire all'applicazione che lo richiede un tempo di trasferimento inferiore a un valore specificato dall'applicazione stessa.

Sicurezza • Un protocollo di trasporto può fornire alle applicazioni anche servizi di sicurezza, in modo da garantire che la comunicazione tra due processi sia incomprensibile ad altri, anche qualora il flusso dei dati sia in qualche modo captato anche da altri processi. Il servizio principale fornito è la **crittografia**, ma esistono anche altri metodi, che verranno discussi sul sito Web associato al presente volume.

Esistono anche altri servizi che il livello di trasporto può mettere a disposizione del livello applicazione. Nel modello **ISO/OSI**, molti di essi risiedono al livello di sessione, livello

che nel modello dell'Internet è confluito in parte nel livello 5 e in parte nel livello 4. La loro discussione sarebbe troppo ampia per i nostri scopi, per cui ci limiteremo a descrivere i servizi forniti dai due protocolli di trasporto principali utilizzati sull'Internet (che peraltro sono ormai di utilizzo universale anche sulle reti locali: **TCP** (*Transmission Control Protocol*) e **UDP** (*User Datagram Protocol*).

Servizi TCP • Il protocollo TCP fornisce un servizio *connection-oriented* e un servizio di trasporto affidabile dei dati: vediamo in sintesi in cosa consistono, rimandando la loro descrizione dettagliata al capitolo successivo.

TCP prevede che client e server si scambino informazioni di controllo a livello trasporto prima che inizi il flusso dei messaggi a livello applicazione. La procedura iniziale di **handshake** (letteralmente *stretta di mano*) allerta i due host in modo che si preparino a gestire il flusso di pacchetti: al termine di essa, si è stabilita una connessione TCP tra i socket dei due processi. La connessione è in modalità **full duplex**, ossia i due host possono inviarsi messaggi nei due sensi contemporaneamente. Quando un'applicazione termina l'invio di messaggi, deve chiudere la connessione TCP esplicitamente: non basta infatti che il flusso si interrompa, affinché si abbia il termine della connessione.

I processi in comunicazione possono fare affidamento su TCP per la consegna di tutti i dati, senza errori e nel corretto ordine: quando un lato del processo passa un flusso di dati al proprio socket, TCP garantisce che il socket dell'altro lato riceva il flusso senza dati mancanti né duplicati e nella stessa sequenza con cui sono stati inviati.

Servizi UDP • Al contrario di TCP, UDP è **connectionless**: non fornisce handshake, non controlla la persistenza della connessione, non garantisce un trasferimento dati affidabile e non controlla la sequenzialità di arrivo dei pacchetti. In compenso è molto leggero e viene perciò utilizzato normalmente dalla telefonia Internet, che gestisce questi parametri con meccanismi interni, e dal DNS, di cui parleremo più avanti.

TCP e UDP non forniscono alcun servizio di gestione dei ritardi e del throughput, ma ciò non significa che applicazioni sensibili ai ritardi e alla disponibilità di banda non possano risiedere sull'Internet; anzi, oggi una buona parte delle applicazioni più usate ricade proprio in questa categoria. Ciò è possibile perché le applicazioni sono state progettate per far fronte a questa mancanza di garanzie.

Protocolli di livello applicazione • Sono i protocolli che definiscono come i processi di un'applicazione, girando su host diversi, si scambiano messaggi. In particolare, un protocollo a livello applicazione definisce:

- la tipologia dei messaggi scambiati (ad esempio messaggi di richiesta e di risposta)
- la sintassi dei vari tipi di messaggi (ad esempio i campi in cui è suddiviso il messaggio e come i campi sono delimitati)
- la semantica dei campi, ossia il significato dell'informazione in essi contenuta
- le regole che determinano come e quando un processo invia messaggi e risponde a messaggi entranti

Alcuni protocolli a livello applicazione sono definiti dallo **IETF (glossario)** in documenti chiamati **RFC** e sono quindi pubblici: nel seguito, descriveremo i più importanti tra questi.

2 HTTP

Nella prima parte del corso abbiamo appreso che HTTP è il protocollo sviluppato per instaurare il meccanismo di richiesta-risposta tra il client e il Web server che contiene le pagine ipertestuali. Andando più in profondità, possiamo dire che **HTTP è un protocollo a livello**

applicazione, che definisce la struttura dei messaggi scambiati tra client e server e il modo con cui questi messaggi vengono scambiati: esistono, quindi, un *lato client* e un *lato server* del protocollo.

Il lato client ha, ovviamente, lo scopo di richiedere a un server una pagina Web, basandosi sull'**URL** che è stata inserita. Ricordiamo qui che l'URL è costituita da tre parti, come si vede in figura 3, e che la parte gestita da HTTP è la terza, ossia l'identificatore di pagina, mentre la seconda è di competenza del client DNS (ne parleremo nel paragrafo successivo).



Figura **3**
Struttura di una URL.

Una volta che il processo DNS ha messo in contatto client e server, HTTP instaura una connessione TCP con il server sulla porta 80, tramite i rispettivi socket; una volta che HTTP ha spedito il messaggio attraverso il socket, questo non è più sotto il suo controllo, ma sotto quello di TCP: in pratica, il messaggio è passato dal livello superiore (livello applicazione) a quello immediatamente inferiore (livello trasporto). Come abbiamo visto sopra, l'utilizzo di TCP garantisce che la richiesta spedita al server e la relativa risposta giungano intatte al destinatario: errori di comunicazione, infatti, precluderebbero da una parte l'identificazione corretta della pagina richiesta e, dall'altra, l'impossibilità di ricostruire correttamente la pagina stessa.

È importante notare che il server invia le pagine richieste senza registrare alcuna informazione relativa al client: se dopo qualche secondo il client ripete la stessa richiesta, il server si limita a soddisfarla di nuovo, senza ricordare di avere appena compiuto la stessa operazione. Poiché HTTP lato server non registra alcuna informazione riguardo ai client, questo processo si definisce *stateless*, parola intraducibile in italiano, ma che potremmo rendere come "senza memoria".

Notiamo anche che l'architettura utilizzata è realmente client-server, perché corrisponde alla definizione data sopra: un Web server è sempre attivo, ha un indirizzo fisso e deve poter soddisfare le richieste di un numero altissimo di browser.

Persistenza delle connessioni • In molte applicazioni di rete che viaggiano sull'Internet, client e server comunicano per periodi piuttosto lunghi, nei quali il client invia una serie di richieste e il server provvede a soddisfarle; secondo l'applicazione che le genera, queste richieste possono essere inviate una dietro l'altra, periodicamente (cioè a intervalli regolari) oppure in modo intermittente cioè non cadenzato. Quando il processo di comunicazione accede a TCP, può instaurare la connessione in due modi diversi: nel primo, ogni coppia richiesta/risposta transita su una connessione TCP separata; nel secondo, tutte le coppie vengono spedite sulla stessa connessione TCP. Nel primo caso, si dice che l'applicazione usa *connessioni non persistenti*, nel secondo che *usa connessioni persistenti*.

HTTP utilizza per default connessioni persistenti, ma può essere configurato anche per sfruttare l'altra modalità.

HTTP con connessione non persistente • Supponiamo di voler scaricare una pagina Web, per esempio `www.ubuntu.com/desktop/features`. Analizziamo il processo passo a passo:

1) il processo HTTP client attiva una connessione TCP con il server `www.ubuntu.com` sulla porta 80, che è la porta di default per HTTP; associati alla connessione, ci sono un socket sul client e uno sul server
2) il client HTTP invia al server un messaggio di richiesta contenente l'identificatore della pagina richiesta (nel nostro caso `/desktop/features`)

3) il server HTTP riceve il messaggio attraverso il proprio socket, recupera l'oggetto `/desktop/features` dalla memoria, lo incapsula in un messaggio HTTP e, attraverso il socket, lo invia al client
4) il processo HTTP server ordina a TCP di chiudere la connessione. TCP lo esegue solo quando è certo che il client ha ricevuto il messaggio intatto
5) il client HTTP riceve il messaggio di risposta e la connessione TCP termina. Il messaggio indica che l'oggetto incapsulato è un file HTML: il browser estrae il file e lo legge, trovando i riferimenti, che altro non sono che ulteriori URL, degli altri oggetti (ad esempio immagini) che risiedono nel server e che sono necessari per completare la visualizzazione della pagina.

I primi quattro passi vengono ripetuti per ognuno dei riferimenti trovati.

Notiamo che HTTP non ha nulla a che vedere con il modo con cui i browser visualizzano la pagina HTML: è solo il protocollo di comunicazione che permette il trasferimento degli oggetti tra server e client.

Se analizziamo il processo dal punto di vista delle connessioni TCP, dato che i riferimenti contenuti nel file HTML sono 10, HTTP instaurerà complessivamente 11 connessioni: ciò non significa, tuttavia, che esse saranno attivate una dopo l'altra, perché i browser delle ultime generazioni sono in grado di gestire sino a 10 connessioni TCP contemporaneamente.

Facciamo ora un calcolo approssimato del tempo necessario a portare termine una richiesta/risposta HTTP. Definiamo, innanzitutto il tempo di andata e ritorno **RTT** (*Round Trip Time*), che è il tempo impiegato da un pacchetto di piccole dimensioni per andare dal client al server e ritornare dal server al client. Come abbiamo visto nel paragrafo precedente, RTT include i ritardi di propagazione, di accodamento e di processo nei vari router coinvolti.

Per ogni richiesta, sono necessari due RTT: il primo per attivare la connessione TCP, il secondo per inviare la richiesta e iniziare la trasmissione del file; a questo si aggiunge, ovviamente, il tempo di trasferimento del file, che dipende essenzialmente dalla sua dimensione e dalla banda disponibile.

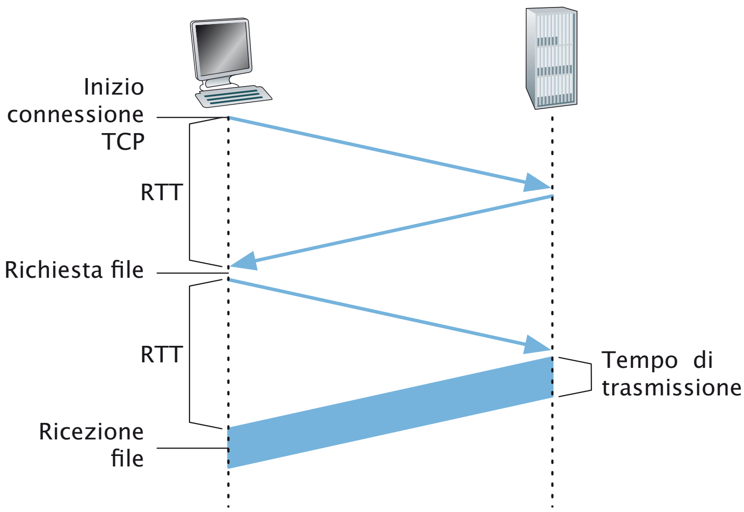


Figura **4**
Tempo necessario a portare a termine una richiesta/risposta HTTP.

HTTP con connessione persistente • Le connessioni non persistenti, oltre a richiedere ogni volta un RTT per instaurarsi e uno per effettuare la richiesta, presentano ulteriori svantaggi, dovuti al fatto che ogni nuova connessione necessita l'allocazione dei buffer TCP e la gestione relativa: ciò può rappresentare un grosso onere per il server, che deve gestire contemporaneamente centinaia di connessioni. Nelle connessioni persistenti (punto 4 dell'elenco precedente) il server non invia a TCP la richiesta di chiusura, in modo tale che le successive richieste provenienti dal client possano essere inviate sulla stessa connessione: le richieste possono, quindi, essere inviate una dietro l'altra, senza attendere risposta, così come gli oggetti possono essere inviati dal server senza tempi morti intermedi dovuti a risposte a richieste in attesa, con un processo chiamato **pipelining**.

Tipicamente, il server HTTP chiude la connessione quando questa non è utilizzata per un certo tempo (la cui durata è configurabile nelle impostazioni del server stesso): per default, HTTP utilizza le connessioni persistenti con il pipelining.

Codici di stato • Nel messaggio di risposta che il processo HTTP server invia al client

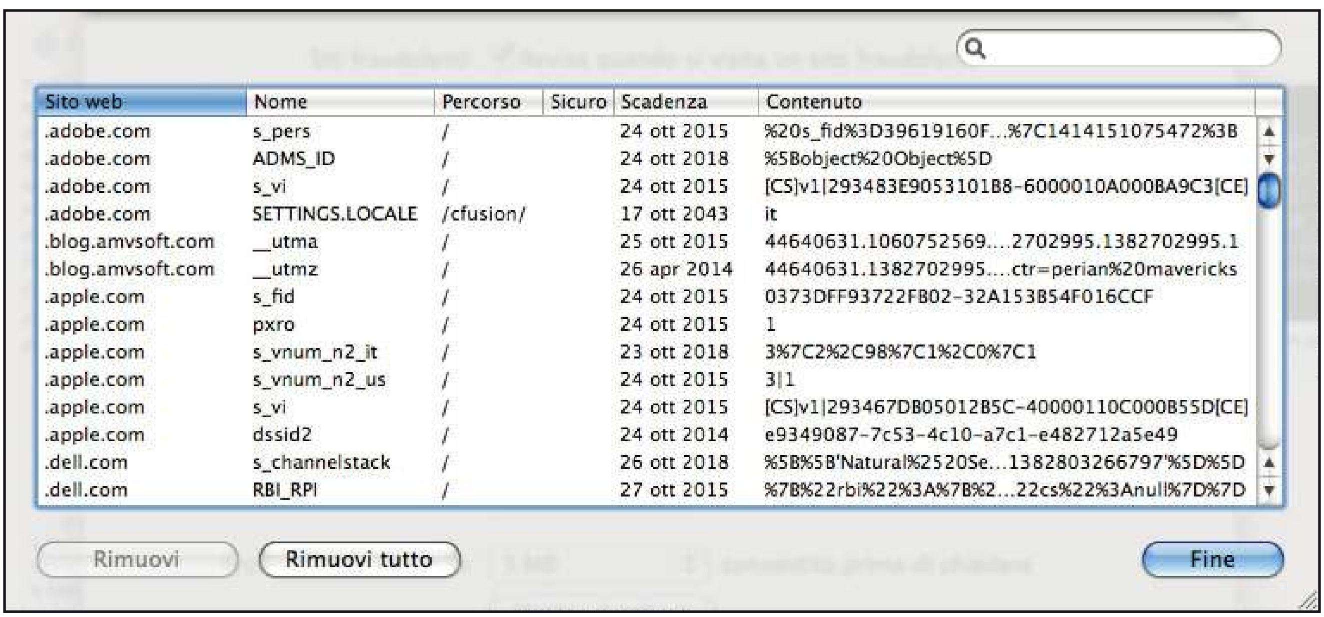
esistono due campi utilizzati per informare il client sul risultato della richiesta e su cosa si deve aspettare nel seguito: il primo fornisce un codice numerico, il secondo una frase esplicativa del codice. Esistono molti codici di stato, ma alcuni si ritrovano molto spesso quando il server rileva un errore nella risposta:

`200 OK`	La richiesta ha avuto successo e l'informazione è stata inviata in risposta: l'utente vede la pagina caricarsi)
`301 Moved Permanently`	L'oggetto richiesto è stato spostato permanentemente. La nuova URL è specificata nel messaggio, e il browser la carica automaticamente: l'utente vede cambiare l'indirizzo nel browser e la pagina caricarsi
`400 Bad Request`	La richiesta non è stata compresa dal server: l'utente non vede accadere nulla
`404 Not Found`	L'oggetto richiesto non è presente sul server: l'utente, di solito, vede una pagina bianca con il codice e la scritta di stato
`505 HTTP Version Not Supported`	La versione del protocollo HTTP usata dal client non è supportata dal server: l'utente, di solito, vede una pagina bianca con il codice e la scritta di stato

Cookie • Abbiamo detto che un server HTTP è *stateless*, ossia non tiene registrazione delle interazioni con i client e gestisce ogni richiesta solo sulla base delle informazioni in essa contenute: in questo modo è possibile minimizzare il tempo di risposta del server, consentendogli di gestire un gran numero di richieste contemporanee.

Tuttavia, a volte è desiderabile che un sito sia in grado di identificare gli utenti, sia per poter limitare gli accessi, sia per poter scegliere i contenuti della pagina sulla base dell'identità dell'utente. Per ottenere questo risultato, il protocollo HTTP usa i **cookie** (letteralmente *biscotti*), che permettono ai siti di tenere traccia degli utenti. Contrariamente a quanto è spesso affermato, i cookie non sono programmi, ma solo righe di testo che il server inserisce nella risposta in una posizione predefinita, e che il browser appende come riga di codice all'interno di un file specifico.

Figura **5**
Struttura dei cookie.



Sito web	Nome	Percorso	Sicuro	Scadenza	Contenuto
.adobe.com	s_pers	/		24 ott 2015	%20s_fid%3D39619160F...%7C1414151075472%3B
.adobe.com	ADMS_ID	/		24 ott 2018	%5Bobject%20Object%5D
.adobe.com	s_vi	/		24 ott 2015	[CS]v1|293483E9053101B8-6000010A000BA9C3[CE]
.adobe.com	SETTINGS.LOCALE	/cfusion/		17 ott 2043	it
.blog.amvsoft.com	__utma	/		25 ott 2015	44640631.1060752569....2702995.1382702995.1
.blog.amvsoft.com	__utmz	/		26 apr 2014	44640631.1382702995....ctr=perian%20mavericks
.apple.com	s_fid	/		24 ott 2015	0373DFF93722FB02-32A153B54F016CCF
.apple.com	pxro	/		24 ott 2015	1
.apple.com	s_vnum_n2_it	/		23 ott 2018	3%7C2%2C98%7C1%2C0%7C1
.apple.com	s_vnum_n2_us	/		24 ott 2015	3|1
.apple.com	s_vi	/		24 ott 2015	[CS]v1|293467DB05012B5C-40000110C000B55D[CE]
.apple.com	dssid2	/		24 ott 2014	e9349087-7c53-4c10-a7c1-e482712a5e49
.dell.com	s_channelstack	/		26 ott 2018	%5B%5B'Natural%2520Se...1382803266797'%5D%5D
.dell.com	RBI_RPI	/		27 ott 2015	%7B%22rbi%22%3A%7B%2...22cs%22%3Anull%7D%7D

Rimuovi   Rimuovi tutto   Fine

All’interno della riga troviamo generalmente sette campi:

- *nome del cookie* (che è ovviamente un campo obbligatorio)
- *nome di dominio del* server che ha inviato il cookie
- *percorso* (non obbligatorio): se esiste, indica una specifica pagina cui il cookie fa riferimento

Questi due ultimi campi indicano al browser che il cookie deve essere inviato al server solo per il dominio e il percorso indicati.

- *scadenza*: indica quando il cookie perderà di valore, e sarà quindi cancellato. Oltre che una data, la scadenza può essere definita come un periodo, oppure può essere definita come **Never**, cioè non avvenire mai
- *sicuro*: indica se il cookie va trasmesso criptato con HTTPS
- *modalità di accesso*: se richiesto, rende invisibile il cookie a linguaggi diversi da HTML, quali ad esempio Javascript
- *contenuto del cookie*: è una stringa di testo di varia lunghezza, che raccoglie le informazioni generate dal server a propria futura memoria

I diversi utilizzi dei cookie sono, principalmente:

- permettere il login a un sito solo a specifici utenti
- personalizzare una pagina Web secondo i desideri dell’utente
- riempire il carrello della spesa virtuale nei siti commerciali
- tracciare i percorsi dell’utente durante la sua navigazione sul Web

I cookie sono spesso utilizzati per registrare i percorsi degli utenti all’interno di un sito, con lo scopo di verificare quali pagine sono di preferenza lette e quali trascurate, fornendo al Webmaster indicazioni utili per migliorare l’usabilità del sito stesso.

Come si vede, alcuni utilizzi dei cookie sono positivi, altri meno: ciò ha portato ad ampie discussioni sul loro impatto nei riguardi della privacy degli utenti, anche in considerazione della possibilità di un uso illegale da parte dei pirati informatici per carpire dati significativi.

A2-01
Web cache

3 FTP, SMTP, POP3, IMAP

Non è possibile descrivere tutti i protocolli di livello applicazione utilizzabili su una rete, la maggior parte dei quali, peraltro, viene utilizzata per applicazioni molto particolari. Dopo aver parlato di HTTP, esamineremo brevemente i più utilizzati dalle applicazioni diverse dal World Wide Web: ***FTP***, ***SMTP***, ***POP3*** e ***IMAP***.

FTP • Benché HTTP sia in grado di trasferire file tra due host, esiste un altro protocollo, **FTP** (*File Transfer Protocol*) dedicato specificamente a questo utilizzo ma dotato di logica e struttura diverse. Un client FTP è molto più semplice e limitato di un browser, dato che il suo unico scopo è permettere l’interconnessione tra client e server per lo scambio dei file.

Per attivare la connessione, l'utente inserisce l'indirizzo del server FTP nel client, che provvede ad instaurare una connessione TCP con il server, che è sempre in ascolto.

Attraverso questa connessione l'utente si identifica con nome e password, che vengono inviati sulla connessione come parte dei comandi FTP. Se il server, sulla base di questi dati, autorizza l'utente, questo può navigare nella parte del file system che gli è stata resa visibile e caricare o scaricare file al suo interno.

La differenza principale tra HTTP e FTP è che quest'ultimo, per trasferire un file, utilizza due connessioni TCP parallele: una connessione di controllo, che abbiamo appena visto, e una connessione dati. Per questa ragione, si dice che FTP invia le informazioni di controllo "*fuori banda*", mentre HTTP lo fa "*in banda*".

Vediamo in dettaglio come funziona il protocollo FTP.

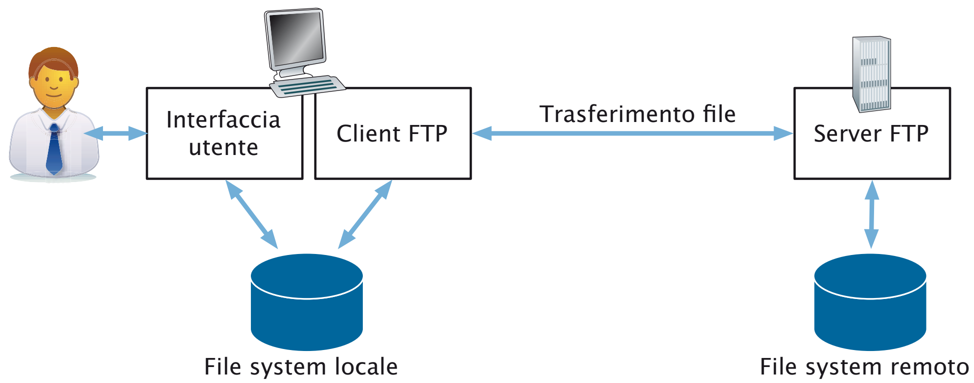


Figura **6**
Funzionamento del protocollo FTP.

Quando un utente inizia una sessione FTP con un host remoto, inserendone l'indirizzo, il client FTP inizia una connessione di controllo con il server FTP sulla porta 21 e invia su di essa nome utente e password. Se il server identifica e autorizza, il client invia - sulla stessa connessione - una serie di comandi che permettono di navigare e di modificare la directory remota all'interno del server.

Nel momento in cui il server riceve il comando di trasferimento di un file, istanzia una connessione TCP verso il client sulla porta 20, attraverso la quale FTP trasferisce uno e un solo file, in una direzione o nell'altra; terminato il trasferimento del file, la connessione dati viene chiusa.

Se, durante la stessa sessione, l'utente vuole trasferire altri file, per ognuno di essi viene aperta una nuova connessione sulla porta 20. Quindi, nel protocollo FTP, la connessione di controllo rimane aperta per la durata dell'intera sessione - quindi è persistente - ma viene creata una nuova connessione non persistente per ogni file che deve essere trasferito.

Il server FTP, a differenza del server HTTP, non può essere stateless, dato che deve mantenere, lungo tutta la sessione, una traccia di dove l'utente si trova all'interno dell'albero della propria directory. Questo tracciamento di ogni sessione riduce in modo significativo il numero di connessioni contemporanee che un server FTP può gestire, rispetto a quelle di un server HTTP.

Comandi e risposte • Come i comandi HTTP, anche quelli FTP sono inviati in formato ASCII, e quindi leggibili direttamente. Ognuno di essi consiste in un gruppo di quattro lettere maiuscole, seguite a volte da argomenti opzionali che lo definiscono compiutamente. I comandi più comuni sono:

`USER nome_utente`	invia il nome utente al server
`PASS password`	invia la password utente al server
`LIST`	chiede al server di inviare una lista dei file presenti nella directory remota corrente. Il server risponde aprendo una connessione dati non persistente, diversa dalla connessione di controllo.
`RETR nome_file`	recupera un file dalla directory corrente. Questo comando provoca l'apertura da parte del server di una connessione dati e attiva il download del file richiesto.

`STOR nome_file`	carica un file locale nella directory remota corrente. Questo comanda l'apertura da parte del server di una connessione dati e l'upload del file da parte del client.

Ogni comando è seguito da una risposta del server verso il client, costituita, come nel protocollo HTTP, da un numero di tre cifre seguito da un messaggio opzionale. Alcune risposte tipiche sono:

```
331 Username OK, password required
123 Data connection already open; transfer starting
425 Can't open data connection
452 Error writing file
```

SMTP • Nella prima parte del corso ci siamo occupati della struttura e del funzionamento del servizio di posta elettronica attraverso l'Internet: in particolare, abbiamo notato come i percorsi di invio e ricezione delle mail siano indipendenti tra loro e possano appoggiarsi a server diversi. Vediamo adesso come opera il protocollo SMTP (che gestisce l'invio della posta, ma non il suo recupero).

Quando si scrive un nuovo messaggio, l'applicazione di posta elettronica provvede a chiuderlo in una busta contenente l'indicazione del destinatario e del mittente e l'oggetto della comunicazione. Quando si spedisce il messaggio, il protocollo SMTP provvede a inviarlo al server cui è abbinato l'account di posta (quando l'account viene creato, l'utente gli associa uno o più server SMTP cui si può collegare per l'invio).

Il server SMTP che ha ricevuto il messaggio si comporta a sua volta come un client e instaura una connessione TCP con il server indicato nell'indirizzo del destinatario. Una volta connesso, SMTP provvede a trasferire il messaggio dal server del mittente a quello del destinatario: nel caso in cui il server del destinatario sia indisponibile, l'operazione viene ritentata più volte.



Figura **7**
Configurazione di un account di posta elettronica.

Dialogo tra server • Per stabilire la connessione e trasferire il messaggio, SMTP utilizza, come fa peraltro HTTP, delle stringhe ASCII contenenti frasi in chiaro, facilmente comprensibili, in testa e in coda al messaggio vero e proprio, inviandole attraverso il proprio socket TCP.

Facciamo un esempio: l'account *daniele@fastweb.it* vuole inviare a *marco@alice.it* il testo "Ci vediamo domani." utilizzando `mailbus.fastweb.it` come server SMTP.

Stabilita la connessione con `alice.it`, i messaggi che i due server si scambiano sono i seguenti:

```
dest:  220 alice.it
mitt:  HELO mailbus.fastweb.it (HELO è un'abbreviazione di hello)
dest:  250 Hello mailbus.fastweb.it, pleased to meet you
mitt:  MAIL FROM: <daniele@fastweb.it>
dest:  250 daniele@fastweb.it ... Sender ok
mitt:  RCPT TO: <marco@alice.it>
dest:  250 <marco@alice.it> ... Recipient ok
mitt:  DATA
dest:  354 Enter mail, end with "." on a line by itself
mitt:  Ci vediamo domani.
```

mitt:	`.`
dest:	`250 Message accepted for delivery`
mitt:	`QUIT`
dest:	`221 alice.it closing connection`

A questo punto la connessione TCP viene chiusa dal server destinatario, nella cui casella di posta è stato registrato il messaggio. Il server SMTP del mittente, invece, non trattiene copia del messaggio.

Notiamo come sia il client, cioè il mittente, a inviare i comandi `HELO`, `MAIL FROM`, `RCPT TO`, `DATA` e `QUIT`, mentre il server ritorna solo messaggi di stato costituiti da numeri di tre cifre, esattamente come accade in HTTP e FTP. Le spiegazioni aggiunte ai comandi e le frasi di cortesia sono opzionali.

Confronto tra HTTP e SMTP • Sia HTTP che SMTP sono utilizzati per trasferire file da un host all'altro: il primo da un Web server a un browser, il secondo da un mail server a un altro. Esistono, tuttavia, importanti differenze tra i due protocolli.

Innanzitutto, HTTP è principalmente un protocollo **pull** (letteralmente *tirare*), perché l'utente "tira fuori" l'informazione dal server secondo le proprie necessità: in particolare, la sessione TCP è iniziata dalla macchina che chiede di ricevere il file.

SMTP, invece, è un protocollo **push** (letteralmente *spingere*), poiché il mittente carica l'informazione sul server del destinatario e la sessione TCP è iniziata dalla macchina che desidera inviare il file.

Una seconda differenza sta nel fatto che il protocollo SMTP richiede che ogni messaggio, incluso il corpo, sia codificato in formato ASCII a 7 bit: se contiene caratteri non conformi a questo formato, ad esempio vocali accentate o codice binario, prima dell'invio questi devono essere ricodificati in ASCII a 7 bit. HTTP, invece, non soffre di questa limitazione. Per aggirarla, è stato strutturato uno standard particolare, chiamato **MIME** (*Multipurpose Internet Mail Extension*), che definisce una serie di ulteriori intestazioni atte a caratterizzare il contenuto dei messaggi.

Una terza differenza riguarda la gestione di quei documenti contenenti, oltre al testo, file di tipo diverso (ad esempio immagini): come abbiamo visto precedentemente, HTTP incapsula ogni singolo oggetto in un proprio messaggio di risposta, mentre SMTP inserisce tutti i file in un unico messaggio.

Protocolli di accesso • Torniamo all'esempio di prima: abbiamo lasciato il messaggio inviato da *daniele@fastweb.it* nel server di posta in cui risiede la casella *marco@alice.it*.

Al destinatario, però, la mail è ancora invisibile: per recuperarla deve utilizzare un protocollo di accesso alla propria casella, che non può essere SMTP, dato che questo è un protocollo push, e quindi non può scaricare alcunché.

Esistono molti possibili protocolli di accesso alla propria casella di mail, ma normalmente ne vengono utilizzati tre: **POP3** (*Post Office Protocol - Version 3*), **IMAP** (*Internet Mail Access Protocol*) e l'onnipresente HTTP, che si utilizza nella **webmail**.

POP3 • Si tratta di un protocollo molto semplice e funzionalmente limitato. Il protocollo attiva l'apertura di una connessione TCP sulla porta 110 del server di posta da cui si devono recuperare i messaggi; una volta stabilita la connessione, POP3 prosegue attraverso tre fasi: *autorizzazione*, *transazione* e *aggiornamento*.
Nella prima fase (autorizzazione) il client invia nome utente e password per autenticarsi presso il server; nella seconda (transazione) recupera i messaggi, li marca o smarca per la cancellazione e ottiene statistiche; nella terza (aggiornamento), dopo la chiusura della connessione TCP, il server provvede a gestire i messaggi scaricati secondo le regole impostate dall'utente.

Durante una connessione POP3, il client invia comandi e il server risponde con due tipi di messaggi: `+OK` (seguito a volte da dati di risposta) oppure `-ERR`.

La fase di autorizzazione consiste in due comandi:

server: `+OK POP3 server ready`
client: `user daniele`
server: `+OK`
client: `pass mia_password`
server: `+OK user successfully logged on`

Terminata questa fase, inizia quella di transazione.

client: `list`
server: `+OK 2 messages`
server: `1 325` *(numero di byte del messaggio)*
server: `2 817` *(numero di byte del messaggio)*
server: `.`
client: `retr 1`
server: *<scarica messaggio 1>*
server: `.`
client: `dele 1`
client: `retr 2`
server: *<scarica messaggio 2>*
server: `.`
client: `dele 2`
client: `quit`
server: `+OK POP3 server signing off`

L'utente può settare il proprio client di posta per chiedere al server di cancellare i messaggi dopo averli scaricati, oppure per tenerli in memoria. In questo caso, il comando `dele` non viene inviato.

Durante la sessione, il server POP3 mantiene traccia di alcune informazioni di stato, ma non le porta tra una sessione e l'altra: tutta la gestione dei messaggi, con la creazione di directory e l'associazione di ulteriori informazioni, la ricerca e la generazione di tag, avviene nel client locale dell'utente.

IMAP • A differenza del POP3, il protocollo IMAP prevede un'interazione molto maggiore tra client e server dato che i messaggi non vengono scaricati sul client in modo permanente (a meno di una specifica richiesta) e tutte le operazioni appena descritte avvengono sul server.

Un server IMAP associa ogni messaggio in arrivo alla cartella INBOX (cartella di entrata) del destinatario, che, a sua volta, può spostarlo in altre cartelle di sua creazione, può leggerlo, cancellarlo, scaricare eventuali allegati o effettuare ricerche nella propria directory secondo i criteri più opportuni: in pratica, fa sul server tutto ciò che un client di posta fa localmente quando usa POP3.

Per questa ragione, è necessario che IMAP tenga traccia dello stato del client tra le varie sessioni, dovendo ricordare la configurazione della directory, la posizione dei messaggi e i vari tag a essi associati. Ne deriva che il numero di comandi disponibili in IMAP è molto superiore a quello di POP3 e che l'implementazione del protocollo sul server è molto più complessa.

Un'altra importante caratteristica di IMAP è la possibilità da parte del client di ottenere solo alcuni componenti dei messaggi, ad esempio la testata o l'indicazione degli allegati. In questo modo l'utente può scegliere se leggere subito per intero il messaggio o meno, cosa che si dimostra utile soprattutto quando la banda disponibile è scarsa.

A2-02
Webmail

4 DNS

Nella prima parte del corso abbiamo descritto il DNS (*Domain Name System*) e le basi del suo funzionamento; adesso entreremo più in profondità nella sua struttura e nelle modalità con cui opera.

Il DNS, in realtà, non è solo un database distribuito con struttura gerarchica, ma è anche **un protocollo di livello applicazione, che permette agli host di interrogare il database stesso**. Il protocollo DNS viene comunemente impiegato da altri protocolli dello stesso livello, inclusi HTTP, FTP e SMTP, per tradurre i nomi di dominio inseriti dagli utenti in indirizzi IP.

Consideriamo cosa accade quando un browser, ossia un client HTTP, richiede un indirizzo, ad esempio `www.zanichelli.it/ricerca`. Per poter inviare la richiesta al server `www.zanichelli.it`, il browser deve innanzitutto ottenerne l'indirizzo IP, che permetterà, come vedremo nel capitolo successivo, il routing della richiesta sino alla connessione fisica del server alla rete. La sequenza di azioni è questa:

1) il browser estrae il nome di dominio dall'indirizzo inserito, partendo dalla doppia barra iniziale e terminando alla prima barra successiva (nell'esempio, estrae `www.zanichelli.it`)

2) il browser passa la stringa al lato client dell'applicazione DNS (che gira sul suo stesso host)

3) utilizzando il protocollo UDP, il client DNS invia la query a un server DNS attraverso la porta 53

4) il server DNS restituisce al client l'indirizzo IP associato al nome di dominio inviato

5) il client DNS passa l'indirizzo al browser, che istanzia una connessione TCP al processo HTTP server localizzato sulla porta 80 dell'host corrispondente all'indirizzo IP

Come funziona il DNS • Guardando la sequenza precedente, si nota come, dal punto di vista dell'applicazione che invoca il servizio DNS, questo sia una scatola nera che fornisce un servizio di traduzione semplice e diretto. In realtà, l'interno di questa scatola nera è complesso, poiché consiste in un gran numero di server distribuiti su tutto il globo, oltre ad un protocollo che specifica come server e client DNS comunicano tra loro.

Il motivo per cui il database è distribuito su un così gran numero di server è facilmente comprensibile.

Una struttura centralizzata, infatti, presenterebbe una serie di inconvenienti sostanzialmente insolubili:

- essendo unica, in caso di guasti il servizio diventerebbe indisponibile, bloccando l'intera Internet
- dovrebbe gestire contemporaneamente centinaia di milioni di richieste, e quindi avrebbe tempi di risposta inaccettabili
- le richieste da parte dei client dovrebbero viaggiare su traiettorie lunghissime e complicate, e intaserebbero i nodi, incrementando ulteriormente i tempi di risposta
- un database centralizzato, oltre ai problemi connessi con la sua dimensione, dovrebbe gestire un'enorme flusso di aggiornamenti, registrare ogni nuovo host che si presenta in rete.

Per queste ragioni, ***la struttura del DNS non può che essere distribuita su un gran numero di server, ciascuno dei quali contiene solo una piccola parte del database.***

Appare scontata la scelta di organizzare i server stessi in una struttura gerarchica, che permette di focalizzarne ulteriormente il lavoro e di diminuire le dimensioni dei singoli database.

In prima approssimazione, esistono tre classi di server DNS:

- i **Root server** (*server radice*)
- i **Top Level Domain server** (*server di dominio di primo livello*)
- gli **Authoritative server** (*server autorevoli*)

I *root server* contengono il database dei *server TLD*, che a loro volta contengono i database dei *server autorevoli* relativi ai domini di livello inferiore che fanno capo al loro dominio di primo livello.

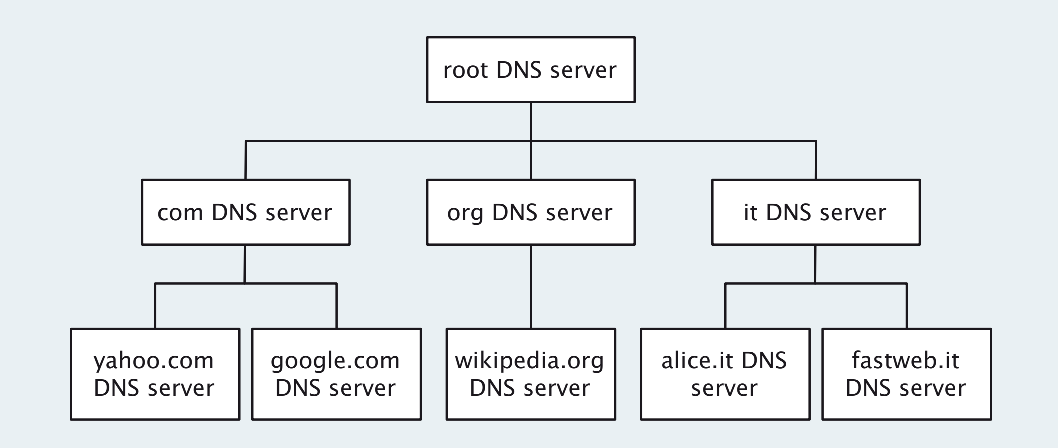


Figura **8**
Gerarchia dei server DNS.

Se un client vuole contattare, ad esempio, *www.amazon.com*, manda la richiesta a un root server, che gli risponde con l'indirizzo IP di un server TLD che gestisce i domini di tipo *.com*. Il client, allora, contatta il server TLD, che gli risponde con un ulteriore IP relativo al server autorevole che gestisce il dominio di secondo livello *amazon*; infine, il client contatta il server autorevole, che gli invia l'indirizzo IP del sito di Amazon.

Ovviamente, nella realtà il processo è più complicato, sia perché esistono più livelli di server autorevoli, sia perché, operando in questo modo, i server radice dovrebbero gestire *tutte* le richieste di *tutti* i client attivi in rete.

Root server • Esistono 13 root DNS server, etichettati dalla A alla M, a cui corrispondono altrettante organizzazioni che li gestiscono. Molte di queste organizzazioni possiedono più sedi, anche in nazioni diverse, per cui il numero di root server è molto più alto: nel 2013 ne erano installati complessivamente 247, sparsi in tutto il mondo, ma localizzati principalmente negli USA e in Europa.

TLD server • Questi server sono responsabili della gestione dei domini di primo livello e sono gestiti da organizzazioni diverse, secondo il tipo di dominio. Le estensioni nazionali, quali *.it*, *.fr*, *.uk*, dipendono direttamente da organizzazioni governative delle singole nazioni. I domini generici, quali *.com*, *.org*, *.net*, *.biz* sono gestiti da organizzazioni con sede in USA, autorizzate a operare dalla **IANA** (*Internet Assigned Number Authority*). Alcuni domini particolari, infine, sono sponsorizzati da specifiche associazioni.

Authoritative server • Ogni organizzazione che controlla host accessibili pubblicamente da Internet, come Web server e mail server, deve rendere pubblici anche i record DNS che permettono di mappare i loro nomi di dominio in indirizzi IP: queste informazioni sono contenute nei propri server autorevoli.
Le grandi organizzazioni internazionali e molte università gestiscono direttamente i propri server. Nella maggior parte dei casi, invece, le informazioni sono raccolte presso i data center dei vari ISP a cui i proprietari dei domini si appoggiano.

Default name server • Per evitare la congestione della rete a causa delle eccessive connessioni del DNS, presso gli ISP sono installati i cosiddetti server DNS *locali*, comunemente chiamati *default name server*, cui i vari host che fanno capo all'ISP si collegano per qualunque richiesta. Se andiamo a vedere la configurazione di rete di un PC connesso a Internet, troviamo un pannello DNS che indica sia il dominio di ricerca, sia l'indirizzo IP del default name server.



Figura **9**
Dominio di ricerca e indirizzo del server locale.

Facciamo un esempio.

L'host *daniele.poli.it* vuole connettersi al server *info.cs.umass.edu*. Supponendo che sia presente un DNS locale, l'host si connette automaticamente a questo (nel nostro caso si chiama *dns.poli.it)*: è questo server che si connette con il root server e riceve gli indirizzi IP dei TLD server che gestiscono il dominio *.edu*.

Avuti gli indirizzi, l'host si connette con uno di questi server, che gli invia l'indirizzo del server autorevole che gestisce il dominio *umass*, al quale a sua volta l'host invia la query. Il server autorevole consultato, allora, invia l'indirizzo corrispondente a *info.cs.umass.edu* al DNS locale, il quale lo gira all'host che ha iniziato la query.

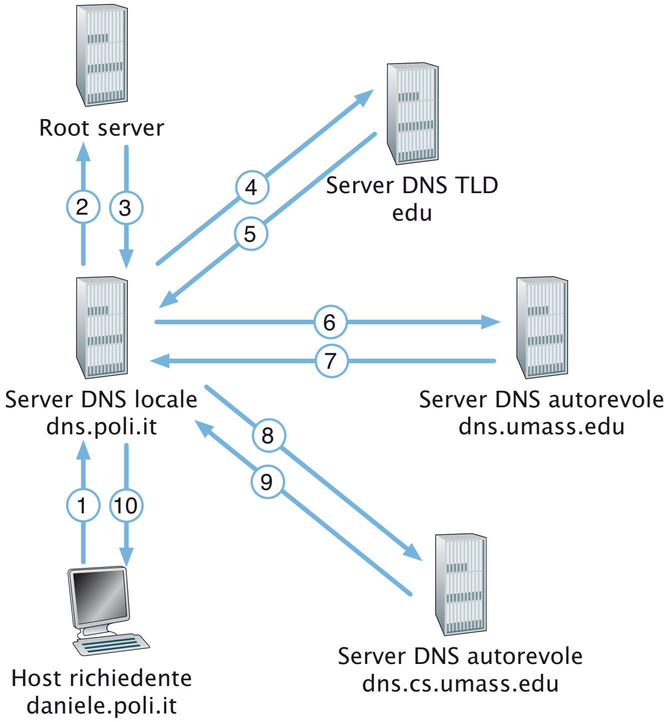


Figura **10**
Interazione tra i server DNS.

La cosa potrebbe essere ulteriormente complicata se l'Università del Massachussets avesse un DNS locale per ogni dipartimento, il server autorevole *dns.umass.edu* non riuscirebbe a risolvere l'indirizzo richiesto, e quindi restituirebbe a *dns.poli.it* l'IP relativo all'ulteriore server autorevole *dns.cs.umass.edu*, cioè quello in grado di risolvere la query.

Osservando il percorso fatto, vediamo come siano presenti due tipi di query: una *ricorsiva*, che si instaura tra l'host *daniele.poli.it* e il server DNS locale *dns.poli.it*; l'altra, *iterativa* in tutti gli altri casi.

La prima query è ricorsiva perché *daniele.poli.it* chiede a *dns.poli.it* di ottenere l'indirizzo per suo conto; le altre sono iterative, perché tutte le risposte sono reindirizzate direttamente a *dns.poli.it*. In pratica, questo è il percorso normalmente utilizzato, ma in teoria tutte le query potrebbero essere di tipo ricorsivo.

DNS caching • Nel processo appena descritto, il numero di connessioni richieste per portarlo a termine è lo stesso che si avrebbe se l'ISP a cui l'host richiedente fa capo non possedesse un default name server.

La caratteristica che permette un drastico miglioramento della situazione è la possibilità da parte dei server di memorizzare localmente gli indirizzi che ricevono, con un'operazione di **caching** (dall'inglese *to cache, nascondere*) che è fondamentale per il funzionamento del sistema.

Per tornare al nostro esempio, quando il server *dns.poli.it* riceve per la prima volta l'IP corrispondente a *info.cs.umass.edu*, provvede a registrarlo nella propria *cache*, in modo che, quando riceve da un host la stessa richiesta, provvede a inoltrare direttamente l'IP senza dover effettuare di nuovo tutta la serie delle query.

Registrar • Sino a ora abbiamo parlato di come i dati vengono recuperati dal database DNS: ora vediamo come essi vengono inseriti.

Se vogliamo creare un nostro dominio, dobbiamo rivolgerci a qualcuno che sia in grado di verificare che il nome da noi scelto non sia già registrato da qualcun altro e, se non lo è, di inserirlo nel database DNS. Esistono a questo scopo delle strutture commerciali, chiamate **Registrar**, che svolgono questi compiti in cambio di un canone annuale, generalmente molto ridotto.

Oltre al nome di dominio, è necessario fornire al Registrar anche nome e indirizzo IP del nostro authoritative server, che il Registrar provvede a inserire nel server TLD corrispondente al dominio di primo livello che abbiamo scelto.

In realtà, nella maggior parte dei casi il nostro sito sarà ospitato presso qualche ISP, che di solito funge anche da Registrar: in questo modo nome e indirizzo saranno quelli dell'ISP.

5 Applicazioni Peer to Peer

A2-03
Applicazioni Peer to Peer

Concetti essenziali

- La comunicazione tra due entità in rete può avvenire secondo modalità client-server oppure peer-to-peer. Esistono anche sistemi che utilizzano un'architettura mista.
- I programmi che girano sugli host sono detti processi.
- Per comunicare tra loro due processi devono utilizzare i servizi messi a disposizione dalla rete sottostante.
- Un processo accede al canale di comunicazione attraverso un socket.
- I diversi protocolli di comunicazione possono essere analizzati in termini di affidabilità, throughput, ritardo e sicurezza.
- Se un protocollo garantisce che i dati trasmessi siano ricevuti completamente e correttamente a destinazione, esso viene definito affidabile.
- Il throughput è il ritmo al quale un processo mittente può inviare dati al processo ricevente.
- Vi sono applicazioni per le quali è indispensabile che il ritardo tra il momento della trasmissione e quello della ricezione sia inferiore a un valore dato.
- Se una comunicazione non deve poter essere comprensibile a terzi, è necessario operare sulle caratteristiche di sicurezza del protocollo.
- Un protocollo a livello applicazione definisce il tipo dei messaggi scambiati, la sintassi dei messaggi, la semantica dei diversi campi e le regole sull'invio dei messaggi e sulla risposta agli stessi.
- Si può comunicare attraverso processi HTTP sia in modalità persistente che non persistente.
- I cookie sono righe di testo che il server inserisce nella risposta in una posizione prefissata e che il browser appende come riga all'interno di un file specifico.
- Il protocollo FTP permette la connessione tra client e server solo per lo scambio di file.
- Il protocollo SMTP gestisce l'invio della posta elettronica ma non il suo recupero.
- I protocolli POP3 e IMAP permettono, con modalità diverse, il recupero dei messaggi di posta.

Test

1 Dire se le seguenti affermazioni sono vere o false.

L'architettura a strati delle reti permette:

A di sviluppare applicazioni indipendentemente dalla gestione della rete V F
B di scambiare tra loro i server e i client durante le fasi di manutenzione V F
C la generazione di siti Web senza dover rispettare alcuna regola V F
D una distribuzione più rapida delle informazioni attraverso l'Internet V F

2 Dire se le seguenti affermazioni sono vere o false.

L'architettura secondo la quale due host comunicano tra loro con dipendenza minima o nulla da un server è detta:

A pipe-to-pipe V F
B point-to-point V F
C peer-to-peer V F
D pear-to-pear V F

3 Dire se le seguenti affermazioni sono vere o false.

In una comunicazione in rete, i processi possono inviare e ricevere messaggi attraverso un'interfaccia con il livello inferiore, detta:

A bucket V F
B socket V F
C rocket V F
D packet V F

4 Dire se le seguenti affermazioni sono vere o false.

Una rete affidabile garantisce:

A che la velocità di comunicazione sia sempre la massima consentita dal mezzo fisico di trasporto V F
B che non si superi mai la lunghezza massima consentita per i messaggi V F
C che i dati trasmessi siano ricevuti completamente e correttamente a destinazione V F
D che ogni pacchetto inviato sia crittografato almeno a 128 bit V F

5 Dire se le seguenti affermazioni sono vere o false.

Il throughput di una rete esprime:

A il ritardo medio tra l'istante di trasmissione di un

pacchetto e quello della sua ricezione a destinazione V F

B l'anticipo con cui arriva l'ultimo pacchetto inviato rispetto al momento previsto in via teorica V F

C il numero minimo di Mb/sec garantiti dal tipo di fibra ottica usata per l'infrastruttura V F

D la velocità del flusso con cui due processi, attraverso la rete, possono comunicare V F

6 **Dire se le seguenti affermazioni sono vere o false.**

La procedura con cui due processi istanziano una comunicazione nel modello di servizio TCP prende il nome di:

A handshake V F

B milkshake V F

C handover V F

D earthquake V F

7 **Dire se le seguenti affermazioni sono vere o false.**

Il protocollo HTTP instaura il meccanismo di richiesta - risposta tra:

A host e centro assistenza V F

B database e pagine Web V F

C client e utente V F

D client e Web server V F

8 **Dire se le seguenti affermazioni sono vere o false.**

Il protocollo HTTP può avere connessioni:

A inalterate V F

B non persistenti V F

C provvisorie V F

D persistenti V F

9 **Dire se le seguenti affermazioni sono vere o false.**

Attraverso i cookie il protocollo HTTP:

A prende il controllo del client e regolarmente lo danneggia V F

B impedisce ai pirati informatici l'accesso alle applicazioni V F

C tiene traccia degli utenti V F

D verifica l'effettiva identità di chi siede alla tastiera V F

10 **Dire se le seguenti affermazioni sono vere o false.**

FTP significa:

A File Transfer Protocol V F

B Fast Track Policy V F

C Find Two Ports V F

D Forward To Police V F

11 **Dire se le seguenti affermazioni sono vere o false.**

Il protocollo SMTP gestisce:

A l'invio della posta elettronica V F

B l'archiviazione della posta elettronica V F

C il recupero della posta elettronica V F

D la periodica eliminazione della posta elettronica V F

12 **Dire se le seguenti affermazioni sono vere o false.**

I seguenti sono protocolli di accesso alla posta elettronica:

A PAP2 V F

B PAYPAL V F

C POP3 V F

D PIN4 V F

13 **Dire se le seguenti affermazioni sono vere o false.**

Il Registrar è:

A la somma che ogni anno il proprietario di un sito deve versare a Microsoft per garantirsi di rimanere unico proprietario del sito e del suo contenuto V F

B un'autorità statunitense che verifica che i contenuti dei siti creati dagli utenti registrati non contengano materiale offensivo, criminale, illegale, ecc. V F

C il nome commerciale del sistema adottato da Google per gestire gli utenti V F

D una struttura aziendale che gestisce i nomi dei domini, impedisce la registrazione di duplicati e inserisce il nome del dominio nel DNS V F

3 Il livello trasporto

Il capitolo, necessariamente anch'esso molto tecnico, conduce gli allievi all'analisi del livello trasporto, dimostrando, attraverso una serie di esempi pratici, come non sia possibile definire un solo protocollo a questo livello. I protocolli cardine di questo livello, TCP e UDP, sono affrontati in modo approfondito. Viene introdotto UDP, che, se per certi aspetti può sembrare inferiore a TCP a un'analisi superficiale, si rivela uno strumento potentissimo, per non dire indispensabile, in situazioni che si presentano con grande frequenza nel mondo delle comunicazioni. Il concetto di multiplexing, sia per TCP che per UDP, insieme a un'analisi delle comunicazioni su canali full-duplex, sono presentati con esempi insieme a una trattazione dettagliata della gestione dei timeout e della risoluzione, a livello trasporto, del problema della congestione.

1 Servizi del livello trasporto

Nel capitolo precedente abbiamo visto le caratteristiche principali dei servizi che il livello trasporto deve fornire ai protocolli del livello superiore, e abbiamo accennato ai due principali protocolli utilizzati sull'Internet: TCP e UDP. Esistono altri protocolli a livello trasporto, utilizzati per scopi particolari: ne sono in fase di sviluppo altri, con lo scopo di unire i pregi di TCP e UDP, ma la quasi totalità delle reti, comprese le LAN, utilizza questi ultimi e nel prossimo futuro non si prevedono cambiamenti.

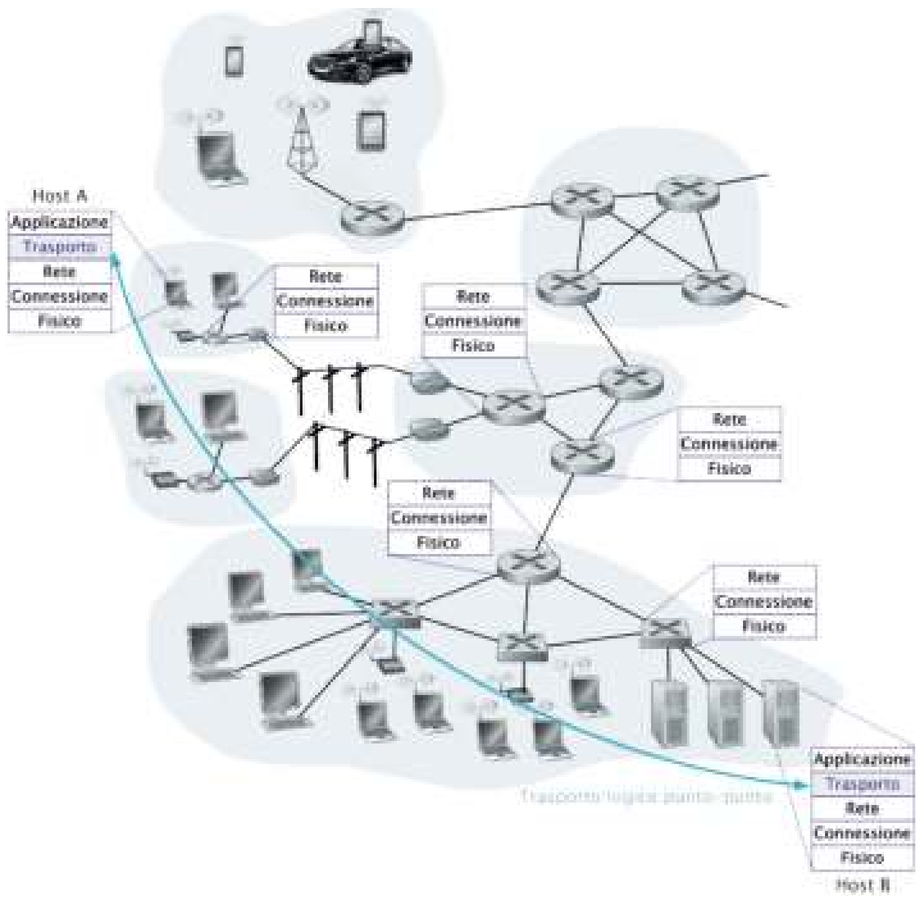


Figura **1**
Comunicazione logica tra processi.

Un protocollo a livello trasporto fornisce una comunicazione logica tra processi che girano su due diversi host: dal punto di vista dell'applicazione sugli host, i due processi sono connessi direttamente tra loro, anche se in realtà possono trovarsi ai lati opposti del pianeta ed essere connessi da un gran numero di nodi e da svariati tipi di connessione. I processi utilizzano la comunicazione logica fornita dal livello trasporto per scambiarsi messaggi, senza doversi preoccupare dei dettagli dell'infrastruttura fisica utilizzata per trasportarli.

La figura 1 illustra la nozione di comunicazione logica.

Come si vede, i protocolli a livello trasporto sono implementati negli host e non nei router della rete. Dal lato mittente, il livello trasporto converte i messaggi che riceve dal livello superiore in segmenti, frammentando i dati inviati dalle applicazioni, se necessario, in blocchi di minori dimensioni e aggiungendo a ogni blocco un header. Il livello trasporto passa i segmenti al sottostante livello rete, che a sua volta incapsula il segmento in un datagram e lo invia a destinazione.È importante notare che i router interagiscono solo con i campi di livello rete presenti nel datagram, e non esaminano i campi a livello trasporto incapsulati nel datagram. Dal lato destinatario, il livello rete estrae il segmento dal datagram e lo passa al livello superiore, che provvede a processarlo, in modo da rendere il blocco di dati disponibile all'applicazione che deve riceverlo.

Relazione tra livello trasporto e livello rete • I protocolli a livello trasporto, come abbiamo detto, forniscono una comunicazione logica tra processi che girano su due host diversi, mentre i protocolli a livello rete forniscono comunicazione logica tra gli host: si tratta di una distinzione sottile, ma fondamentale per comprendere il funzionamento delle reti.

Per comprenderla meglio, usiamo un'analogia.

Poniamo che in due città lontane tra loro, ad esempio Milano e Londra, vi siano due palazzi condominiali i cui abitanti comunichino abitualmente tra loro per lettera. Ognuno scrive il proprio messaggio e lo infila nella propria casella in portineria.

Il portinaio preleva i plichi dalle caselle e li consegna al servizio postale, nella persona del postino che giornalmente passa a ritirare la posta.

Il servizio postale, attraverso vari uffici e mezzi di trasporto, provvede a far giungere i plichi alla portineria dell'altro condominio, dove il portinaio li distribuisce nelle varie caselle, dalle quali i destinatari possono prelevarli e leggerli.

I condòmini corrispondono alle applicazioni, i plichi ai messaggi generati, le caselle ai socket, il portinaio al livello trasporto e il servizio postale al livello rete (ma anche ai livelli inferiori).

Dal punto di vista dei condòmini, il servizio postale è rappresentato dal portinaio, che è l'unico con il quale hanno un rapporto diretto, anche se la consegna avviene attraverso le caselle di posta, e non hanno necessità di sapere quello che avviene a valle.

Allo stesso modo, il portinaio risiede nel condominio, non conosce affatto i meccanismi di trasporto del servizio postale e non deve spostarsi fuori dell'edificio per compiere il proprio lavoro.

Multiplexing • Presso qualunque host destinatario, il livello trasporto riceve i segmenti dal sottostante livello rete e ha il compito di consegnare i dati in essi contenuti all'applicazione appropriata tra quelle che girano sull'host. Se, ad esempio, stiamo inviando e ricevendo mail, scaricando una pagina Web e trasferendo un file, abbiamo sicuramente aperti contemporaneamente quattro processi di rete: un POP3, un SMTP, un HTTP e un FTP. Quando il livello trasporto riceve i dati dal livello inferiore, si trova nella necessità di indirizzarli a uno di questi processi.

Abbiamo visto nel capitolo precedente che la comunicazione con le applicazioni avviene attraverso uno o più socket, quindi il livello trasporto nell'host destinatario non consegna i dati direttamente al processo, ma al socket intermedio: poiché, come nell'esempio appena fatto, possono esserci più socket attivi contemporaneamente, ognuno deve disporre di un identificatore univoco.

Quando il livello trasporto riceve i blocchi di dati da diverse applicazioni - quindi attraverso più socket - esso provvede a incapsularli nei segmenti aggiungendo un header che contiene le informazioni relative al socket di provenienza e a quello di destinazione, e li passa al livello rete perché li instradi sino a destinazione.

Questo processo si chiama ***multiplexing***, perché il flusso che ne deriva può essere diretto a diversi destinatari e, sullo stesso destinatario, a diverse applicazioni.

Quando un segmento viene consegnato al livello trasporto del destinatario, il protocollo legge le informazioni relative al socket, spacchetta i dati e li consegna al socket indicato: questo processo inverso si chiama ***demultiplexing***.

La sequenza di operazioni è descritta nella figura 2.

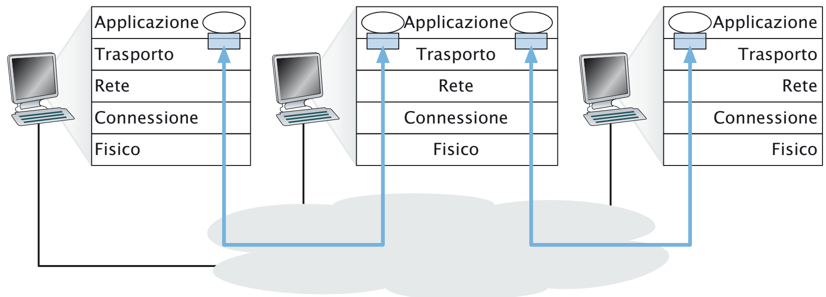


Figura **2**
Multiplexing e demultiplexing.

Notiamo che il livello trasporto dell'host centrale deve demultiplare segmenti che provengono dal livello sottostante verso due distinti processi e, nello stesso tempo, poiché la comunicazione è bidirezionale, multiplare verso due host diversi i dati che gli vengono passati dal livello superiore. Abbiamo descritto il processo di multiplexing/demultiplexing per il livello trasporto, ma è importante notare che, in una rete, questo processo avviene tutte le volte che più protocolli a un livello accedono a un solo protocollo al livello inferiore.

Gli identificatori univoci utilizzati per riconoscere i socket di partenza e destinazione consistono nel numero di porta utilizzato da un processo per comunicare con il livello sottostante, e sono inseriti dal protocollo di trasporto nei primi due campi dell'header che costituisce la prima parte di un segmento. Indipendentemente dal protocollo di trasporto utilizzato, l'header di un segmento ha la struttura di figura 3: le differenze tra un segmento UDP e un segmento TCP verranno analizzate nel seguito.

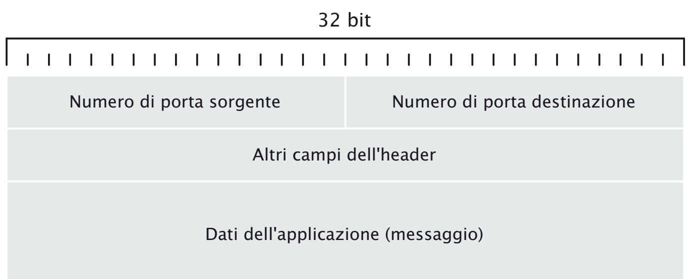


Figura **3**
Numeri di porta sorgente e di porta destinazione in un segmento.

I due campi si chiamano *porta sorgente* e *porta destinatario* e contengono ciascuno un numero di 16 bit - quindi compreso tra 0 e 65535 - detto *numero di porta*, che identifica univocamente il socket di uno specifico protocollo a livello applicazione.

I numeri di porta compresi tra 0 e 1023 sono detti *numeri noti* (*well-known port numbers*) e sono riservati a protocolli, appunto, noti: ad esempio, HTTP utilizza la porta 80, FTP la porta 21.

Multiplexing UDP • In modalità *connectionless*, quando viene creato un socket da un protocollo a livello applicazione, il livello trasporto verifica innanzitutto che abbia un numero di porta: in caso contrario, provvede automaticamente ad assegnargliene uno compreso tra 1024 e 65535, scegliendo tra quelli che non sono già assegnati all'interno dell'host.

Tipicamente, accade che le applicazioni che girano su un server abbiano già assegnato internamente il numero di porta (normalmente un numero noto), mentre dal lato client l'applicazione consente al livello trasporto di assegnare il numero di porta in modo automatico e trasparente.

Supponiamo che un processo nell'host A, con porta 19157, voglia inviare un blocco di dati a un processo con porta 46428 sull'host B. Il livello trasporto nell'host A crea uno o più segmenti che contengono i dati, il numero della porta sorgente (19157) e il numero della porta destinazione (46428), e li passa al livello rete.

Il livello rete incapsula il segmento in un datagram IP e cerca di consegnarlo all'host B. Se il segmento arriva all'host B, il livello trasporto al suo interno esamina il numero della porta destinazione e, se esiste un socket associato a quel numero, lo consegna.

Poiché nell'host B girano più processi, probabilmente il livello trasporto deve effettuare un demultiplexing dei segmenti in arrivo, provvedendo a smistarli ai vari socket attivi.

È importante notare che un socket UDP è pienamente identificato dall'indirizzo IP dell'host e dal numero di porta destinazione: se due segmenti hanno diversi indirizzi IP del mittente e/o diverse porte sorgenti, ma hanno lo stesso indirizzo IP del destinatario e la stessa porta destinazione, entrambi saranno indirizzati allo stesso processo sulla stessa porta.

L'host B, per rispondere, prende il numero di porta sorgente dell'host A (19157) e lo inserisce nel segmento di risposta come numero di porta destinatario, così come il protocollo al livello inferiore utilizza l'indirizzo IP del mittente come indirizzo di destinazione (figura 4).

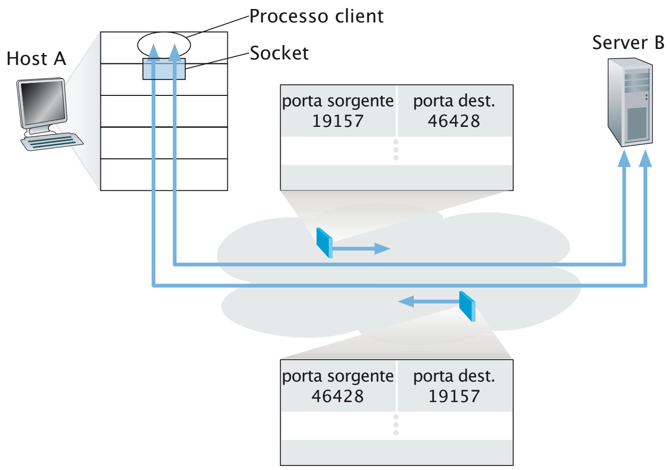


Figura **4**
Inversione dei numeri di porta sorgente e di porta destinazione.

Multiplexing TCP • In modalità *connection oriented*, il socket e il modo con cui viene stabilita la connessione presentano alcune differenze.

Per prima cosa, un socket TCP è identificato da quattro valori: indirizzo IP mittente, numero di porta sorgente, indirizzo IP destinatario, numero di porta destinazione. Quando un segmento giunge a un host, questo utilizza tutti i quattro valori per demultiplarlo verso il socket appropriato.

In particolare, diversamente da UDP, segmenti TCP in ingresso con IP mittente o numero di porta sorgente diversi sono indirizzati a due diversi socket.

Per comprendere le ragioni di questo diverso comportamento, vediamo cosa accade quando un server, su cui gira una singola applicazione, deve ricevere richieste da host e da processi diversi, facendo riferimento alla figura 5:

- L'applicazione ha un cosiddetto ***welcoming socket*** (*socket di benvenuto*) TCP, su cui rimane in attesa di richieste di connessione dai client TCP, sulla porta 12000
- Il client, il cui processo sa di questa porta, crea un socket e invia una richiesta di connessione alla porta 12000 del server. La richiesta di connessione è un segmento particolare, di cui parleremo più avanti, contenente il numero di porta destinazione 12000 e un numero di porta sorgente scelto dal client: nel nostro caso 26145, 7532 e 26145
- Quando l'applicazione che gira sul server riceve la richiesta di connessione sulla porta

12000, localizza il processo che sta aspettando una connessione su quella porta e provvede a generare un nuovo socket
- TCP sul server annota i quattro valori identificativi del segmento di richiesta (indirizzo IP mittente, numero di porta sorgente, il proprio indirizzo IP, numero di porta destinazione). Il socket appena creato viene identificato con questi quattro valori: tutti i segmenti successivi che contengono questi valori saranno indirizzati allo specifico socket. Ciò significa che un segmento proveniente dallo stesso host client, ma da un socket diverso, causa la generazione di un nuovo socket sul server.

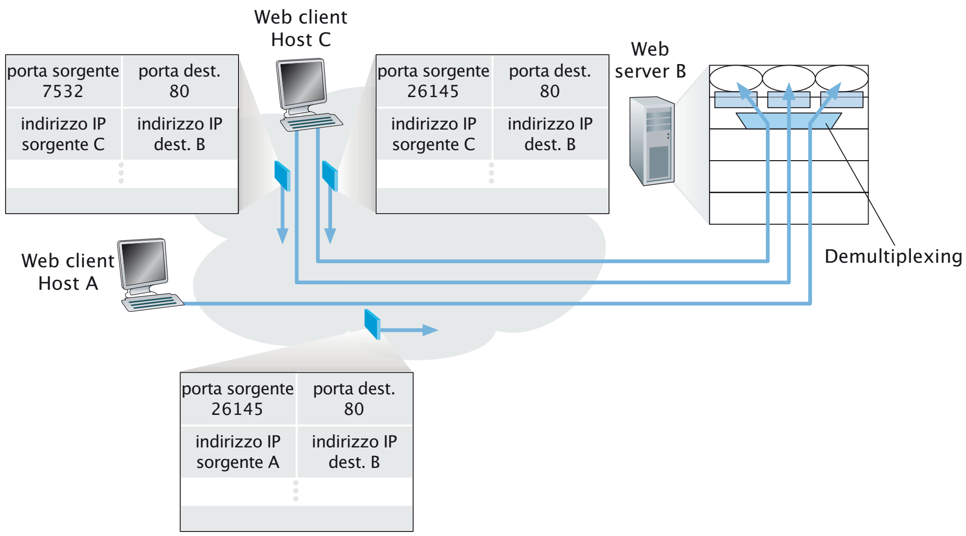


Figura **5**
Uso del numero di porta sorgente e dell'indirizzo IP per riconoscere i processi client.

L'applicazione sull'host server può normalmente gestire molti socket TCP, in cui ogni socket è collegato a un processo e identificato dai quattro valori identificativi dei segmenti.

In questo modo, il server è in grado di demultiplare molte connessioni contemporanee verso la stessa applicazione senza commettere errori. Nel caso limite in cui due client scelgano lo stesso numero di porta sorgente, il protocollo a livello trasporto è in grado di smistare correttamente i segmenti, utilizzando il diverso indirizzo IP in essi contenuto.

2 UDP

UDP è un protocollo estremamente semplice, che fa il minimo indispensabile per effettuare il trasporto: oltre alla funzione di multiplexing/demultiplexing e a una semplice verifica degli errori, non aggiunge nulla al sottostante IP. In effetti, quando un'applicazione utilizza UDP, in pratica parla quasi direttamente con IP.

UDP prende i dati provenienti dal processo al livello superiore, aggiunge un header contenente le porte sorgente e di destinazione e altri due piccoli campi e passa il segmento risultante al livello rete. Il livello rete lo incapsula in un datagram IP e tenta, con il suo metodo *best effort* - inaffidabile, come vedremo nel capitolo successivo - di consegnare il segmento al destinatario. Se il segmento giunge a destinazione, UDP usa il numero della porta di destinazione per consegnarlo al corretto processo.

Notiamo come con UDP non vi sia handshake tra mittente e destinatario al livello trasporto, prima di procedere all'invio di un segmento.

Controllo di flusso • UDP non fornisce controllo di flusso, diversamente da TCP, come vedremo più avanti. Consideriamo l'invio di un flusso di segmenti dall'host A all'host B tramite UDP: alla ricezione, tipicamente UDP appende i segmenti uno dietro l'altro in un buffer

di dimensioni finite che precede il socket corrispondente, ossia che è immediatamente prima della porta di comunicazione del processo. Il processo legge un intero segmento per volta, prelevandolo dal buffer: se la velocità di ricezione è superiore a quella di lettura, il buffer non è più in grado di accettare ulteriori segmenti in arrivo, che vengono scartati.

Vantaggi di UDP • Benché apparentemente l'utilizzo di TCP, che fornisce un servizio di consegna affidabile, sia sempre preferibile, esistono applicazioni che sono più adatte ad appoggiarsi a UDP, per quattro ragioni:

- *controllo più fine al livello applicazione* di quali dati sono inviati e quando. Sotto UDP, appena il processo passa i dati al livello sottostante, il protocollo li impacchetta in un segmento UDP, che consegna immediatamente al livello rete. TCP, come vedremo, possiede invece un controllo di congestione che rallenta il mittente quando uno o più nodi lungo il percorso soffrono di sovraccarico. TCP, inoltre, continua a ritrasmettere un segmento sino a che il destinatario non ne conferma la ricezione, indipendentemente dal tempo necessario alla consegna affidabile. Poiché le applicazioni in tempo reale spesso richiedono una velocità minima di invio, non vogliono eccessivi ritardi di trasmissione e tollerano una certa perdita di dati, il modello di servizio di TCP non è particolarmente adatto a esse: in questi casi, è preferibile utilizzare UDP e implementare, direttamente nell'applicazione, le funzionalità addizionali che mancano al protocollo a livello trasporto
- *nessuna impostazione della connessione*: come vedremo più avanti, TCP usa un complesso handshake prima di iniziare a trasferire dati. UDP, invece, inizia a trasmettere senza alcun preliminare, per cui non introduce ritardi allo scopo di stabilire una connessione. Questa è, probabilmente, la ragione principale per cui DNS gira sopra UDP: se utilizzasse TCP sarebbe molto più lento. HTTP, invece, si appoggia a TCP, perché l'affidabilità è critica nella trasmissione dei testi, mentre un certo ritardo nel download delle pagine Web è accettabile.
- *nessuno stato di connessione*: TCP mantiene lo stato di connessione negli host: ciò comprende buffer di invio e ricezione, variabili per il controllo della congestione e variabili per l'affidabilità del trasporto. UDP, d'altra parte, non mantiene alcuno stato di connessione e non gestisce alcuna variabile; per questa ragione, un server dedicato a una specifica applicazione può gestire molti più client attivi quando l'applicazione gira sopra UDP anziché TCP.
- *header piccolo*: un segmento TCP ha un header di 20 byte, mentre un segmento UDP ha un header di solo 8 byte.

La tabella 1 elenca le principali applicazioni che utilizzano Internet e il protocollo di trasporto che utilizzano.

Come ci si può aspettare, Web, e-mail e file transfer girano sopra TCP, dato che necessitano

Applicazione	Protocollo a livello applicazione	Protocollo a livello trasporto
Posta elettronica	SMTP	TCP
Accesso remoto	Telnet	TCP
Web	HTTP	TCP
Trasferimento file	FTP	TCP
File server remoto	NFS	Tipicamente UDP
Streaming multimediale	Proprietario	UDP o TCP
Telefonia su Internet	Proprietario	UDP o TCP
Management di rete	SNMP	Tipicamente UDP
Protocollo di routing	RIP	Tipicamente UDP
Nomi di dominio	DNS	Tipicamente UDP

Tabella **1**
Principali applicazioni Internet e protocolli utilizzati.

di un servizio di trasferimento affidabile. UDP è utilizzato, invece, dai processi di aggiornamento delle tabelle di routing a livello rete, di cui parleremo nel prossimo capitolo; dalle applicazioni di gestione della rete, come ***SNMP*** (*Simple Network Management Protocol*), che devono poter girare proprio quando le condizioni della rete sono critiche, ossia quando un trasferimento affidabile e un controllo di congestione sono difficilmente ottenibili; da DNS, che deve occupare poco spazio in rete e ottenere risposte il più velocemente possibile.

Per quanto riguarda le applicazioni multimediali, sono utilizzati entrambi i protocolli, dato che comunque possono tollerare una certa quantità di pacchetti persi e che il trasferimento affidabile non è una necessità critica; inoltre, le applicazioni in tempo reale, come VoIP e videoconferenza, reagiscono malamente al controllo di congestione di TCP.

Tuttavia, l'utilizzo esteso di UDP nelle applicazioni multimediali può causare problemi all'interno della rete. Come abbiamo visto, UDP non ha un controllo di congestione, ma questo è necessario per evitare che la rete entri in uno stato quasi di stallo, in cui poco traffico passa effettivamente: se tutti cominciassero a scaricare video ad alta velocità, ci sarebbe un tale sovraccarico nei router, che ben pochi pacchetti UDP riuscirebbero ad attraversare il percorso dal mittente al destinatario.

Inoltre, l'alto tasso di perdita di pacchetti, indotto dai mittenti UDP senza controllo, causerebbe un drammatico rallentamento dei mittenti TCP che, come vedremo più avanti, regolano la propria velocità in funzione del livello di congestione.

Struttura del segmento • Il segmento UDP ha una struttura molto semplice, come si vede in figura 6: il campo dati è preceduto da un header di soli quattro campi, ognuno lungo 2 byte.

- *Porta sorgente*: indica la porta usata dal processo mittente per passare i dati a UDP

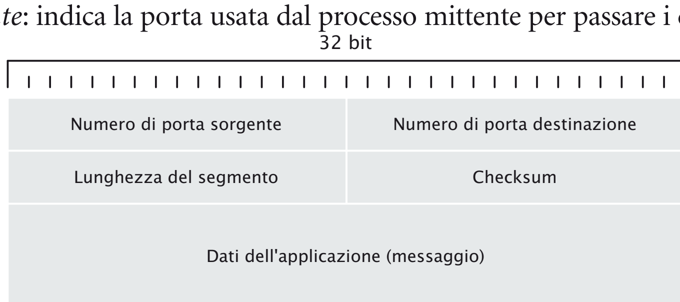


Figura **6**
Struttura del segmento UDP.

- *Porta destinazione*: indica la porta attraverso la quale il processo sull'host destinatario può leggere i dati inviati
- *Lunghezza*: specifica il numero totale di byte del segmento, compreso l'header. È necessario fornire un valore esplicito di lunghezza, perché il campo dati può avere dimensioni variabili da un segmento all'altro
- *Checksum*: campo utilizzato dall'host destinatario per verificare se, durante il trasferimento, sono stati introdotti errori nel segmento. Parleremo dei principi di rilevamento e correzione degli errori nel capitolo 5

Rilevazione degli errori • Possiamo chiederci la ragione per cui UDP genera un checksum per la rilevazione degli errori, quando i principali protocolli di livello 2, come vedremo nel capitolo 5, forniscono lo stesso servizio, con strategie molto più raffinate e performanti.

La ragione è che non esiste garanzia che tutte le connessioni tra mittente e destinatario utilizzino un protocollo dotato di questo servizio; inoltre, anche se i segmenti sono trasferiti correttamente attraverso una connessione, possono avvenire errori durante la permanenza all'interno dei nodi.

Poiché non è possibile garantire né l'affidabilità delle singole connessioni, né il rilevamento degli errori nei singoli nodi, UDP deve fornire un rilevamento a livello trasporto, su una base

punto-punto tra mittente e destinatario.

Dato che il protocollo IP a livello rete si può appoggiare su qualunque protocollo a livello connessione, è necessario che sia il livello trasporto a fornire la rilevazione degli errori come misura di sicurezza.

Notiamo, comunque, che UDP non fornisce alcun servizio di correzione degli errori: alcune implementazioni di UDP scartano semplicemente il segmento danneggiato, altre lo passano all'applicazione soprastante, con un avvertimento.

3 TCP

Abbiamo visto nel capitolo 2 che TCP è detto *connection-oriented* perché, prima che un processo possa iniziare a inviare dati a un altro, questi devono effettuare un *handshake* tra loro, ossia scambiarsi un certo numero di segmenti al fine di stabilire i parametri del successivo trasferimento di dati. Parte di questo processo consiste nell'inizializzazione di molte variabili di stato del TCP, associate con la connessione, di cui parleremo più oltre.

La connessione TCP non è un circuito punto-punto, come nelle reti a commutazione di circuito, né un circuito virtuale, in quanto lo stato della connessione risiede interamente nei due host: il protocollo TCP, infatti, gira solo negli host e non negli elementi intermedi della rete, che ne sono totalmente inconsapevoli e che, quindi, non influiscono sullo stato della connessione.

Una connessione TCP fornisce un servizio full duplex: se si instaura una connessione TCP tra il processo A su un host e il processo B su un altro host, i dati a livello applicazione possono fluire da A e B e contemporaneamente da B ad A.

Inoltre, una connessione TCP è sempre punto-punto, ossia tra un singolo mittente e un singolo destinatario: il ***multicast***, ovvero il trasferimento dati da un singolo mittente a più destinatari, non è possibile utilizzando il protocollo TCP.

Connessione • Per stabilire una connessione TCP, il processo client inizia informando il livello trasporto che vuole connettersi con un processo server; a questo scopo, TCP invia uno speciale segmento al server, che risponde con un altro segmento specifico al quale, a sua volta, il client risponde con un terzo segmento: i primi due segmenti non contengono dati, mentre il terzo può contenerne.

Una volta instaurata la connessione, i due processi possono cominciare a scambiarsi dati: il processo client passa attraverso il socket un flusso di dati che il TCP locale prende in carico e dirige al buffer di invio della connessione. Questo è uno dei buffer che vengono generati durante il processo di handshaking iniziale. Periodicamente, TCP preleva blocchi di dati dal buffer di invio e li passa al livello rete (figura 7).

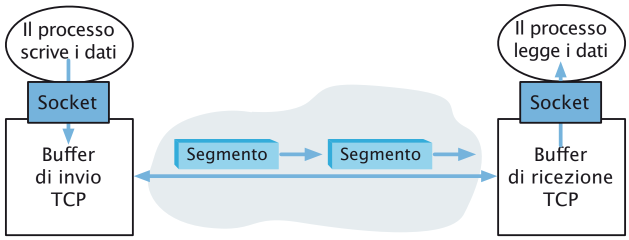


Figura **7**
Buffer TCP di invio e ricezione.

La quantità di dati che può essere prelevata e inserita in un segmento è limitata dalla dimensione massima del segmento stesso, detta ***MSS*** (*Maximum Segment Size*) che, a sua volta, è determinata dalla lunghezza massima del frame a livello connessione, che può essere inviato dall'host mittente, detta ***MTU*** (*Maximum Transmission Unit*).

MSS viene dimensionato in modo che un segmento TCP, incapsulato in un datagram IP, sommato alla lunghezza dell'header TCP/IP sia contenuto in un singolo frame a livello connessione.

TCP accoppia ogni blocco di dati a un header, formando così il segmento, che passa poi al livello rete, dove viene incapsulato in un datagram IP che, a sua volta, è inviato in rete. Quando il processo TCP all'altro estremo riceve un segmento, i dati in esso contenuti vengono caricati nel buffer di ingresso, dal quale l'applicazione legge il flusso attraverso il proprio socket.

Vediamo, quindi, che una connessione TCP consiste in buffer, variabili e una connessione socket a un processo all'interno dei due host, mentre nulla è allocato negli apparati di rete interposti tra loro.

Struttura del segmento TCP • Il segmento TCP consiste in un header, formato da un certo numero di campi, e da un campo dati, la cui massima dimensione è limitata da MSS. Quando TCP invia un grosso file, ad esempio un'immagine, tipicamente divide il flusso dei dati in più blocchi grandi quanto MSS; le applicazioni interattive, invece, spesso trasmettono blocchi di dimensioni inferiori a MSS, richiedendo segmenti di piccole dimensioni.
Vediamo il formato del segmento TCP (figura 8):

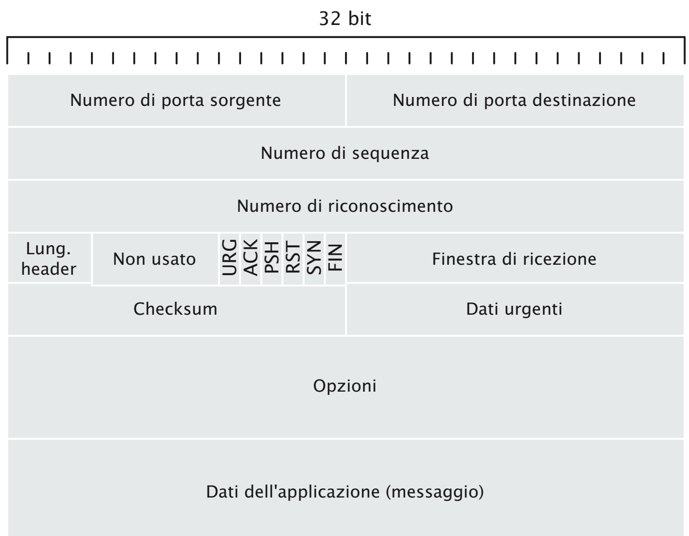


Figura **8**
Struttura del segmento TCP.

- *Porta sorgente*: indica la porta usata dal processo mittente per passare i dati a TCP
- *Porta destinazione*: indica la porta attraverso la quale il processo sull'host destinatario può leggere i dati inviati
- *Numero di sequenza*: serve per implementare il trasferimento affidabile dei dati, e indica il numero del primo byte del segmento all'interno della sequenza del flusso di byte
- *Numero di riconoscimento*: è il numero di sequenza del prossimo segmento che l'host si aspetta di ricevere: questo campo e il precedente verranno descritti più accuratamente in seguito
- *Lunghezza dell'header*: specifica la lunghezza dell'header TCP, che può variare a causa della presenza del campo Opzioni. Tipicamente, questo campo è vuoto e la lunghezza dell'header corrisponde a 20 byte
- *Flag*: è un campo di 6 bit, ciascuno dei quali ha un significato specifico:
  - *URG*: indica che nel segmento sono presenti dati che il mittente considera urgenti
  - *AC*: indica che il valore riportato nel campo riconoscimento è valido, ossia che il segmento contiene l'attestazione che un segmento è stato ricevuto correttamente
  - *PSH*: se questo bit è posto a 1, significa che il destinatario deve passare immediatamente i dati al livello superiore, senza lasciarli nel buffer di ricezione
  - *RST*, *SYN*, *FIN*: questi bit vengono utilizzati per stabilire e chiudere la connessione, come vedremo nel seguito

- *Finestra di ricezione*: serve per il controllo del flusso dati. Indica il numero di byte che il destinatario è disposto ad accettare
- *Checksum*: campo che serve per il controllo degli errori di trasmissione
- *Dati urgenti*: indica la posizione, all'interno del segmento, dell'ultimo byte del blocco di dati urgenti
- *Opzioni*: si tratta di un campo a lunghezza variabile utilizzato quando due host devono negoziare MSS o quando devono essere definiti parametri particolari per connettersi attraverso reti ad alta velocità. Può contenere anche un *time stamp* (letteralmente *marca temporale*) ossia un valore di data e ora con significato particolare
- *Dati*: è il campo che contiene il flusso inviato dal processo a livello applicazione

Numero di sequenza e numero di riconoscimento • Come già detto, si tratta di due campi dell'header che permettono a TCP di realizzare il servizio di trasferimento affidabile dei dati. Poiché TCP vede i dati come una sequenza non strutturata, ma ordinata, i *numeri di sequenza* sono relativi al flusso di byte trasmessi e non alla serie di segmenti in cui questi sono suddivisi.

Facciamo un esempio.

Supponiamo che un processo nell'host A voglia inviare un flusso di dati a un processo nell'host B. TCP nell'host A numera implicitamente ogni byte del flusso, a partire dal primo che gli arriva: se, per semplicità, diciamo che questo consiste in un file di 500 KB e MSS è 1 KB, TCP lo dividerà in 500 segmenti e chiamerà 0 il primo segmento, dato che tale è il numero associato con il primo byte. Il secondo segmento avrà numero di sequenza 1 000, che è il numero associato al suo primo byte, il terzo segmento 2 000 e così via sino all'ultimo segmento che avrà numero di sequenza 499 000.

Consideriamo adesso il *numero di riconoscimento*: poiché TCP è una connessione full duplex, l'host A può ricevere dati dall'host B contemporaneamente al proprio invio, all'interno della stessa connessione. Supponiamo che l'host A abbia ricevuto un segmento contenente i byte da 0 a 535 e che stia per trasmettere a sua volta un segmento.

Poiché la numerazione dei byte è strettamente consequenziale, il successivo segmento inviato dall'host B inizierà con il byte 536, quindi avrà 536 come valore all'interno del campo numero di sequenza.

L'host A metterà, quindi, 536 nel campo numero di riconoscimento del segmento che invia, per dire all'host B che si aspetta che il successivo ricevuto segmento contenga questo valore nel campo numero di sequenza.

Se il segmento successivo ha, invece, un numero di sequenza più alto, ad esempio 900, significa che il segmento contenente i byte da 536 a 899 è andato perso e l'host B ne è informato.

Quando ciò accade - e vedremo nei capitoli successivi che con determinati mezzi trasmissivi si tratta di una situazione molto comune - esistono due possibili strategie per il destinatario, entrambe accettabili secondo le specifiche del protocollo TCP. Con la prima, il segmento che inizia per 900 viene immediatamente scartato dall'host A, in attesa della ritrasmissione del segmento precedente, mentre la seconda possibilità prevede che il segmento venga registrato in un'area specifica del buffer di ingresso: TCP provvede poi a rimontare correttamente la sequenza quando riceve il segmento mancante. Questa seconda opzione è quella utilizzata normalmente nelle implementazioni pratiche, in quanto risulta più efficiente in termini di velocità di trasmissione e occupazione della rete.

Da parte sua, quando l'host B riceve un numero di riconoscimento più basso del numero di sequenza dell'ultimo segmento inviato, sa che uno o più segmenti sono andati persi, quindi provvede alla loro ritrasmissione.

Timeout e RTT • Per evitare di ritrasmettere un segmento che non è andato perso, ma semplicemente è ancora in transito sulla rete, TCP usa un meccanismo di ***timeout***. Il tempo di timeout, ovviamente, deve essere maggiore del tempo di andata e ritorno della connessione, ossia del tempo che intercorre tra l'invio di un segmento e la risposta contenente il

riconoscimento. Abbiamo visto nel capitolo precedente che questo tempo viene detto ***RTT***.

Il calcolo di RTT, tuttavia, non è immediato.

Chiamiamo *RTT campione*, per ogni segmento, il tempo compreso tra quando il segmento viene inviato (ossia passato a IP) e quando viene ricevuto il relativo riconoscimento: poiché *RTT* è per forza maggiore del solo tempo di trasmissione del segmento, se TCP lo misurasse per ogni segmento trasmesso, vi sarebbero più misure in corso contemporaneamente.

Per evitare questo spreco di risorse, TCP effettua una misura alla volta. Ciò significa che, in qualunque istante, *RTT campione* è stimato per uno solo alla volta dei segmenti inviati e non ancora riconosciuti, portando a ottenere un nuovo valore di *RTT campione*, approssimativamente vicino all'*RTT effettivo*.

Ovviamente, *RTT campione* cambia da segmento a segmento, a causa delle congestioni nella rete e del carico del destinatario e, in molti casi, il valore misurato si discosta da quello tipico, che potremmo immaginare come una interpolazione tra i valori degli RTT stimati.

Per queste ragioni, è necessario fare una sorta di media delle misure, che viene chiamata *RTT stimato*.

A ogni nuova misura, TCP aggiorna il valore usando la formula:

$$RTT\ stimato = (1 - \alpha) \cdot RTT\ stimato + \alpha \cdot RTT\ campione$$

dove il fattore α è generalmente posto uguale a 1/8.

Notiamo che, con questa formula, si ottiene una media pesata che dà maggior valore ai campioni recenti rispetto a quelli più vecchi, ossia una *media mobile esponenziale pesata*.

La figura 9 nostra l'andamento tipico di *RTT stimato* rispetto alle misure di *RTT campione*.

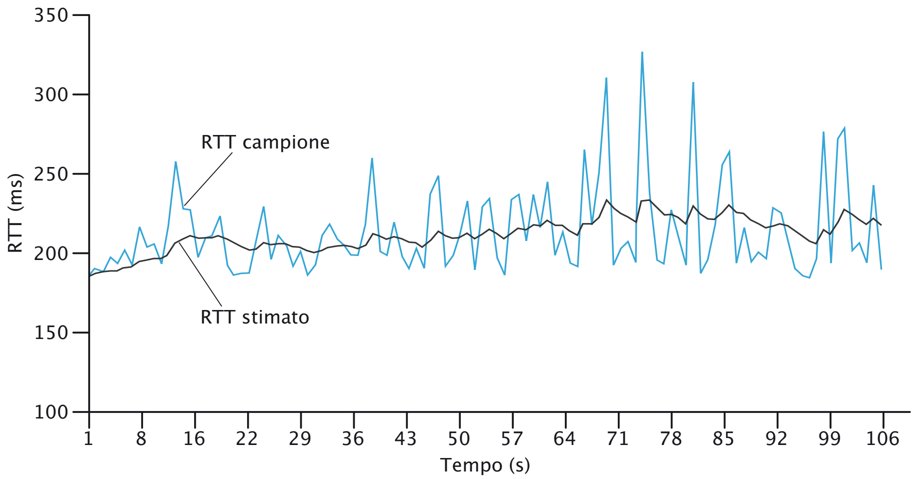


Figura **9**
Andamento tipico di RTT campione e RTT stimato.

È utile, inoltre, avere una misura della variabilità di RTT, definita dal parametro *RTT dev*, che si ottiene con la formula:

$$RTT\ dev = (1 - \beta) \cdot RTT\ dev + \beta \cdot (RTT\ campione - RTT\ stimato)$$

dove β è generalmente 1/4.

Essendo *RTT dev* una media mobile esponenziale pesata della differenza tra *RTT campione* e *RTT stimato*, *RTT dev* è tanto più piccolo, quanto sono minori le fluttuazioni di *RTT campione*.

Avendo calcolato *RTT stimato* e *RTT dev*, TCP deve decidere quale valore scegliere per l'intervallo di timeout: chiaramente, deve essere superiore a *RTT stimato*, altrimenti vi sarebbero ritrasmissioni non necessarie.

Non deve essere, però, troppo maggiore di *RTT stimato* altrimenti, a fronte della perdita di un segmento, TCP non sarebbe in grado di ritrasmetterlo velocemente, rallentando il

trasferimento dei dati: il margine rispetto a *RTT stimato* deve variare in continuazione, adattandosi all'ampiezza delle fluttuazioni di *RTT campione*.

Il metodo utilizzato da TCP per calcolare il valore di timeout è:

$$\text{intervallo di timeout} = RTT\ stimato + 4 \cdot RTT\ dev$$

Trasferimento dati affidabile • Abbiamo visto nel capitolo 1 e vedremo nel capitolo successivo che, nell'Internet, il servizio a livello rete è inaffidabile: IP non garantisce la consegna dei datagram, né l'ordine corretto della consegna stessa, né l'integrità dei dati all'interno del datagram.

Con il servizio IP i datagram possono straripare dai buffer dei router e non giungere mai a destinazione; possono arrivare con sequenze casuali e i bit al loro interno possono essere corrotti: di conseguenza, poiché i segmenti a livello trasporto sono portati attraverso la rete dai datagram IP, possono soffrire anch'essi degli stessi problemi.

TCP realizza un servizio di trasferimento dati affidabile sopra l'inaffidabilità del livello inferiore: il servizio assicura che il flusso di dati che un processo legge dal buffer TCP di ricezione è non corrotto, senza lacune, senza duplicati e in sequenza, ossia che il flusso ricevuto è esattamente identico a quello spedito.

La struttura del mittente TCP è molto complessa, per cui nel seguito utilizzeremo alcune ipotesi semplificative: supporremo che il mittente non sia vincolato dal controllo di congestione, che i blocchi dati siano tutti inferiori al MSS e che il flusso dati avvenga in una sola direzione.

Vi sono tre eventi principali relativi alla trasmissione e ritrasmissione dei dati:

- sono ricevuti dei dati dall'applicazione al livello superiore
- avviene un timeout
- viene ricevuto un riconoscimento (ACK)

Quando accade il primo evento, TCP prende i dati, li incapsula in un segmento e passa il segmento a IP. Nello stesso momento - se il timer di timeout non è già partito per qualche altro segmento - inizializza il timer utilizzando come scadenza la formula:

$$\text{intervallo di timeout} = RTT\ stimato + 4 \cdot RTT\ dev$$

vista nel capoverso precedente.

Il secondo evento è la scadenza del timeout: TCP risponde ritrasmettendo il segmento che ha causato il timeout, dopodiché reinizializza il timer.

Il terzo evento che TCP deve gestire è l'arrivo di un riconoscimento dal destinatario, ossia di un segmento che contiene un valore valido del sottocampo ACK. Quando ciò avviene, TCP confronta il valore del campo *numero di riconoscimento* con una variabile interna, che chiamiamo per comodità *base_invio* e che contiene il numero di sequenza del più vecchio segmento inviato che non sia stato ancora riconosciuto.

TCP utilizza un sistema di riconoscimento cumulativo, per cui l'ACK di un determinato segmento si estende a tutti i segmenti precedenti che non l'hanno ancora ricevuto: se il numero di riconoscimento è maggiore di *base_invio*, tutti i segmenti compresi tra *base_invio* e l'ACK vengono considerati accettati. TCP, quindi, aggiorna *base_invio* al numero di sequenza del primo segmento non riconosciuto e reinizializza il timer di timeout.

Casi pratici • Questa descrizione del funzionamento di TCP è, come abbiamo detto, estremamente semplificata, ma consente di descriverne il funzionamento in alcuni casi pratici che si presentano frequentemente.

Il ***primo caso*** è descritto in figura 10, in cui l'host A invia un segmento all'host B.

Supponiamo che il segmento abbia numero di sequenza 92 e contenga 8 byte di dati: A, quindi, si aspetta di ricevere un segmento ACK con numero di riconoscimento pari a 92 + 8 = 100.

Benché il segmento sia regolarmente ricevuto da B, il suo ACK viene perso nel percorso da

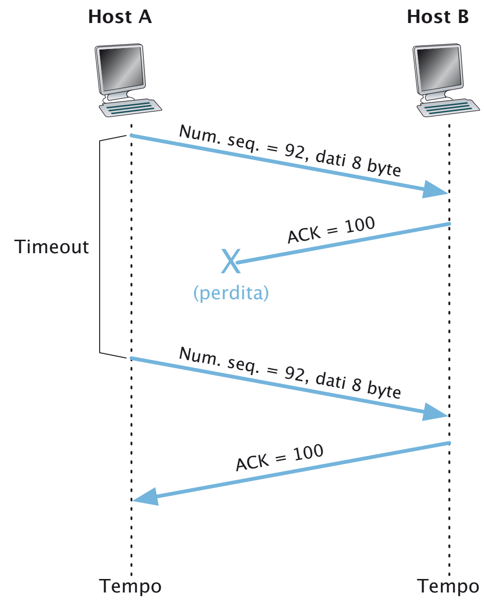


Figura **10**
Ritrasmissione dovuta alla perdita di un ACK.

B ad A, per cui il timeout nell'host A scade e A ritrasmette il segmento.
Quando B riceve la ritrasmissione, constata che il numero di sequenza del segmento corrisponde a quello di un segmento già ricevuto: il TCP nell'host B, quindi, reinvia ad A un nuovo ACK e scarta i byte ricevuti con la ritrasmissione.

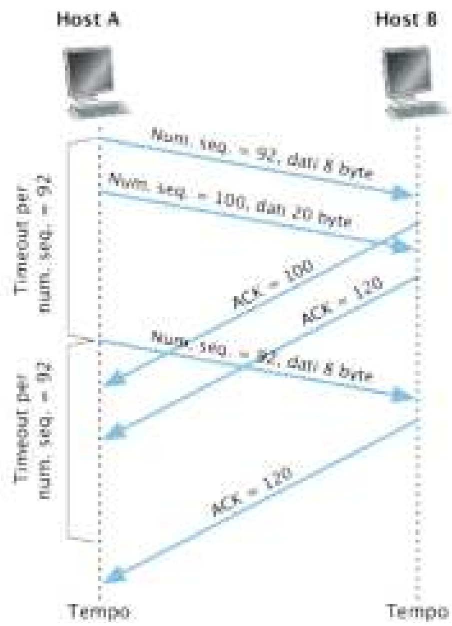


Figura **11**
Il segmento 100 non viene ritrasmesso.

Vediamo il ***secondo caso***, descritto nella figura 11.
L'host A invia due segmenti in sequenza: il primo ha numero di sequenza 92 e contiene 8 byte, il secondo ha, ovviamente, numero di sequenza 100 e contiene 20 byte. Supponiamo che entrambi i segmenti giungano intatti all'host B e che questo risponda con due distinti segmenti di ACK che, quindi, contengono i numeri di riconoscimento 100 e 120, ma che entrambi raggiungano A quando è già avvenuto il timeout.
Al timeout, TCP in A verifica che *base_invio* contiene il numero 92, che corrisponde al più vecchio dei segmenti inviati e lo ritrasmette, riavviando anche il timer. Prima che scada

il timeout della ritrasmissione, A riceve i due ACK di B e verifica che il secondo contiene il numero di riconoscimento 120: a questo punto, TCP sa che anche il secondo segmento è stato ricevuto correttamente e non provvede alla ritrasmissione.

Nel frattempo B ha ricevuto il secondo invio del segmento con numero di sequenza 92, che è già stato ricevuto regolarmente, per cui TCP provvede e scartarlo e a ritrasmettere ad A il segmento con ACK 120. A, a sua volta, avendolo già ricevuto, si limita a ignorarlo, in quanto la variabile *base_invio* è già stata aggiornata.

Il ***terzo caso*** è descritto in figura 12 e presuppone che l'host A invii i due segmenti, come nell'esempio precedente, ma che l'ACK del primo segmento vada perso, mentre quello del

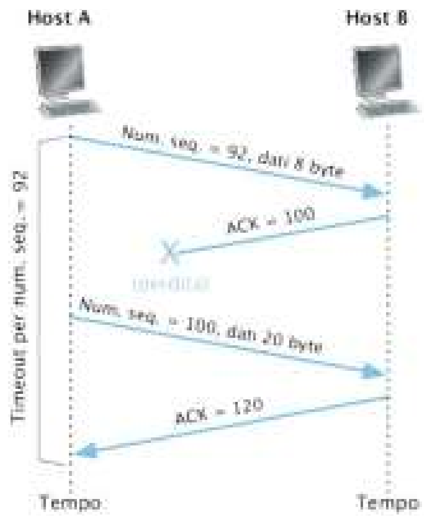


Figura **12**
Un ACK cumulativo evita la ritrasmissione del primo segmento.

secondo giunga regolarmente prima della scadenza del timeout.

A questo punto, TCP in A sa che tutti i byte sino a 119 sono stati regolarmente ricevuti, per cui non effettua nessuna ritrasmissione.

Raddoppio del timeout • In molte implementazioni di TCP, una volta scaduto i timeout, il segmento viene ritrasmesso, ma il nuovo intervallo di timeout viene raddoppiato, anziché essere derivato da RTT.

Se, ad esempio il timeout associato con il più vecchio segmento non riconosciuto è di 1 s, TCP lo ritrasmette dopo questo tempo, ma pone il nuovo timeout a 2 s; nel caso anche la ritrasmissione non riceva un ACK, il terzo invio avrà un timeout di 4 s. Ovviamente, se nel frattempo arriva un ACK relativo a un segmento con numero di sequenza più alto, il processo viene annullato e si torna al timeout calcolato su RTT.

Questo artificio consente un limitato, ma semplice, controllo delle congestioni sulla rete.

La scadenza del timeout, infatti, è molto probabilmente causata da una congestione su qualche nodo, ossia da un'eccessiva quantità di pacchetti che giungono alle code dei router lungo il percorso tra mittente e destinatario: questa situazione causa perdita dei pacchetti o tempi di transito molto lunghi, come vedremo nel capitolo successivo.

In questo caso, la continua ritrasmissione dei segmenti provoca un peggioramento della congestione, mentre l'allungamento del timeout consente ai router di smaltire le code, ricevendo progressivamente sempre meno pacchetti.

Ritrasmissione veloce • Le ritrasmissioni innescate da timeout hanno il problema che la scadenza può essere relativamente lunga: quando un segmento si perde, questo forza il mittente a posporne il reinvio, aumentando così il ritardo punto-punto.

Fortunatamente, il mittente può spesso individuare la perdita di pacchetti prima che scada il timeout, notando quelli che in gergo vengono definiti *ACK duplicati*, ossia riconoscimenti di segmenti che avevano già ricevuto un ACK in precedenza.

Per capire la risposta del mittente a un ACK duplicato, dobbiamo prima comprendere perché il destinatario può inviarne: la tabella riassume il metodo di generazione degli ACK da parte di TCP (tabella 2).

Evento	Azione del TCP destinatario
Arrivo di un segmento in ordine e con il numero di sequenza previsto. Tutti i dati sino al numero di sequenza previsto già riconosciuti.	ACK differito. Attesa sino a 500 ms dell'arrivo di un altro segmento in ordine. Se il segmento non arriva entro 500 ms, invio di ACK.
Arrivo di un segmento in ordine e con il numero di sequenza previsto. Un altro segmento in ordine in attesa della trasmissione di ACK.	Invio immediato di ACK cumulativo, con riconoscimento di entrambi i segmenti.
Arrivo di un segmento fuori ordine e con il numero di sequenza maggiore del previsto. Rilevato un vuoto.	Invio immediato di ACK duplicato, con indicazione del numero di sequenza del successivo byte atteso.
Arrivo di un segmento che riempie in parte o totalmente il vuoto nei dati ricevuti.	Invio immediato di ACK, a condizione che il segmento inizi al punto più basso del vuoto.

Tabella **2**
Generazione degli ACK da parte di TCP.

Quando un TCP destinatario riceve un segmento con un numero di sequenza maggiore di quello atteso, rileva una lacuna nel flusso dei dati, ossia un segmento mancante, che può essere dovuto a perdite o disordine all'interno della rete. Poiché TCP non usa ***NAK***, ossia *riconoscimenti negativi*, il destinatario non può inviare al mittente un messaggio esplicito di non ricezione: per fornire questa informazione, rimanda un riconoscimento dell'ultimo segmento ricevuto nella corretta sequenza, ossia un ACK duplicato.

Poiché il mittente manda spesso un gran numero di segmenti uno in coda all'altro, è probabile che vi siano anche molti ACK duplicati uno dietro l'altro.

Dato il meccanismo di reinvio appena descritto, se il TCP mittente riceve tre volte lo stesso ACK, considera perso il segmento immediatamente successivo: in questo caso, opera quella che viene detta *ritrasmissione veloce*, inviando nuovamente il segmento prima che scada il timeout, come mostrato in figura 13.

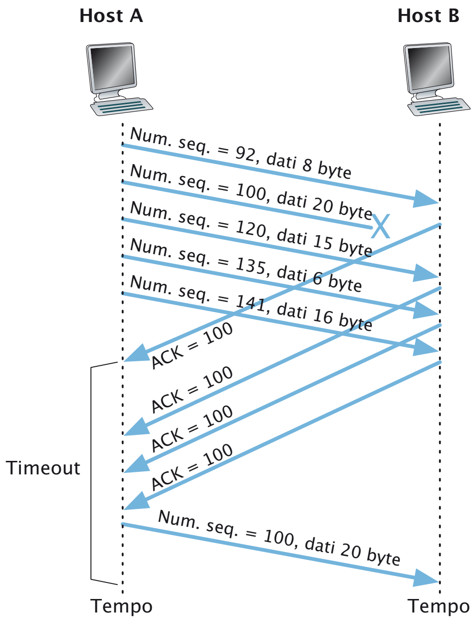


Figura **13**
Ritrasmissione veloce.

Controllo di flusso • Abbiamo visto che gli host da entrambi i lati di una connessione

TCP generano un buffer di ricezione. Quando TCP riceve byte che sono corretti e in sequenza, li colloca nel buffer di ricezione; il processo associato alla connessione legge i dati presenti nel buffer, ma non necessariamente appena arrivano: l'applicazione, infatti, potrebbe essere impegnata in un altro task, e non tentare neppure di leggere i dati, se non molto dopo che sono arrivati.

Se l'applicazione è relativamente lenta nella lettura dei dati, il mittente può facilmente rischiare di saturare il buffer di ricezione, inviando troppi dati e troppo velocemente.

Per evitare questa situazione, TCP fornisce un *servizio di controllo di flusso*, che consiste nell'adeguare la velocità di trasmissione alla velocità di ricezione all'altro capo della connessione. In realtà, come mostrato nel precedente paragrafo, anche il controllo di congestione agisce sulla velocità di trasmissione, ma le ragioni sono completamente diverse.

TCP fornisce il controllo di flusso attraverso una variabile, mantenuta dal destinatario, chiamata *finestra di ricezione*, il cui scopo è quello di fornire al mittente un'indicazione di massima dello spazio disponibile nel buffer di ricezione del destinatario; poiché TCP è full duplex, ognuno dei due host mantiene una finestra di ricezione autonoma.

Ipotizziamo a titolo di esempio che l'host A stia inviando all'host B un grosso file. L'host B alloca un buffer di ricezione per questa connessione - TCP può avere più connessioni contemporanee con più host, per ognuna delle quali genera un buffer dedicato - dalla quale periodicamente l'applicazione relativa legge i dati.

Facendo riferimento alla figura 14, chiamiamo *buffer_ric* la dimensione del buffer e *fin_ric* la variabile *finestra di ricezione*.

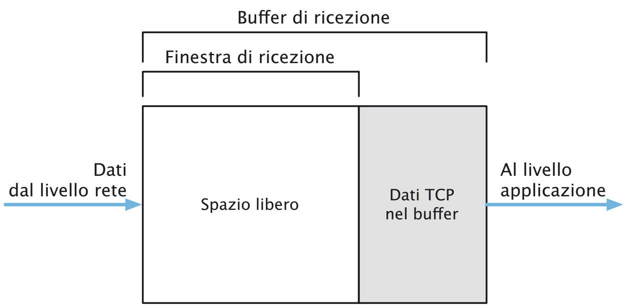


Figura **14**
Buffer di ricezione e finestra di ricezione.

Definiamo due variabili:

- *ultimo_letto*: il numero di sequenza dell'ultimo byte che è stato letto dal processo soprastante e, quindi, prelevato dal buffer
- *ultimo_ricevuto*: il numero di sequenza dell'ultimo byte che è stato ricevuto dalla rete e inserito nel buffer

Poiché TCP non può saturare il buffer, dobbiamo avere:

$$(ultimo_letto - ultimo_ricevuto) \leq buffer_ric$$

e la finestra di ricezione risulta

$$fin_ric = buffer_ric - (ultimo_letto - ultimo_ricevuto)$$

Poiché lo spazio libero cambia nel tempo, *fin_ric* è una grandezza dinamica.

L'host B comunica all'host A quanto spazio libero ha a disposizione nel buffer, inserendo il valore *fin_ric* nel campo finestra di ricezione di ogni segmento che invia ad A. Questi, a propria volta, gestisce altre due variabili:

- *ultimo_inviato*: è il numero di sequenza dell'ultimo byte che TCP ha passato in rete
- *ultimo_riconosciuto*: è il numero di sequenza ricevuto con l'ultimo ACK

Notiamo che la differenza tra *ultimo_inviato* e *ultimo_riconosciuto* rappresenta la quantità di dati inviati da A che si trovano all'interno della rete e non sono ancora giunti a B.

L'host A si assicura di non saturare il buffer di B, facendo in modo che

$$(ultimo_inviato - ultimo_riconosciuto) \leq fin_ric$$

garantendo che durante tutta la durata della connessione non vi siano perdite di dati nel buffer di ricezione di B.

Questo schema funziona benissimo, ma presenta una situazione di paralisi: poniamo che il buffer di B sia pieno (cioè che *fin_ric* = 0), quindi che A non spedisca ulteriori segmenti a B, e che B non abbia nulla da spedire ad A.

A questo punto, poiché non deve spedire né dati né ACK, l'host B non può generare un nuovo segmento con cui informare l'host A che il buffer si sta svuotando, e quindi la comunicazione rimane interrotta, in quanto A non può sapere cosa succede all'altro capo della connessione: si genera, quindi, una situazione di stallo.

Per risolvere il problema, TCP prevede che, quando la finestra di ricezione di B ha dimensione 0, A continui a inviare segmenti contenenti un solo byte. Questi segmenti, quando giungono a B, causano la generazione di un ACK, che contiene anche la nuova dimensione di *fin_ric* e, quindi, il flusso di informazioni nella connessione non viene arrestato.

Gestione della connessione • Supponiamo che un processo che gira su un host client voglia stabilire una connessione con un processo attivo su un secondo host server.

L'applicazione client inizia informando TCP che vuole connettersi a una specifica applicazione sul server; TCP sul client, quindi, procede alla connessione con TCP sul server utilizzando la seguente procedura:

1. TCP lato client invia uno speciale segmento a TCP lato server, che non contiene dati provenienti dal livello applicazione.
   In questo segmento il bit SYN dell'header, che abbiamo visto nella descrizione della struttura dei segmenti, è posto a 1: per questa ragione, il segmento è chiamato *segmento SYN*.
   Inoltre, il client sceglie a caso un numero di sequenza iniziale, che chiamiamo *num_ini_client*, e lo inserisce nel campo *numero di sequenza* del segmento SYN.
   A questo punto il segmento è passato a IP, che lo incapsula in un datagram e lo spedisce in rete.

2. Quando il datagram IP contenente il segmento SYN arriva all' host server - sempre che arrivi - questo provvede ad estrarre il segmento dal datagram, allocare i buffer e le variabili TCP della connessione ed invia un segmento di risposta all'host client.
   Neppure questo segmento contiene dati provenienti dal livello superiore, ma porta con sé tre importanti informazioni, contenute nell'header:
   - il bit SYN è messo a 1
   - il campo numero di riconoscimento contiene (*num_ini_client* + 1)
   - contiene il numero di sequenza iniziale scelto dal server, che chiamiamo *num_ini_server*.

   Questo segmento di risposta è chiamato SYNACK.

3. Alla ricezione del segmento SYNACK, anche TCP del client provvede ad allocare i buffer e le variabili alla connessione.
   L'host client, a questo punto, invia un terzo segmento al server, in cui il campo *numero di riconoscimento* contiene (*num_ini_server* +1) e il bit SYN è a posto a 0, comunicando con ciò di aver accettato la connessione e che questa è regolarmente instaurata.
   Questo terzo segmento può già contenere dati provenienti dal processo superiore.

Una volta completati questi tre passi, i due host possono reciprocamente inviarsi segmenti contenenti dati.

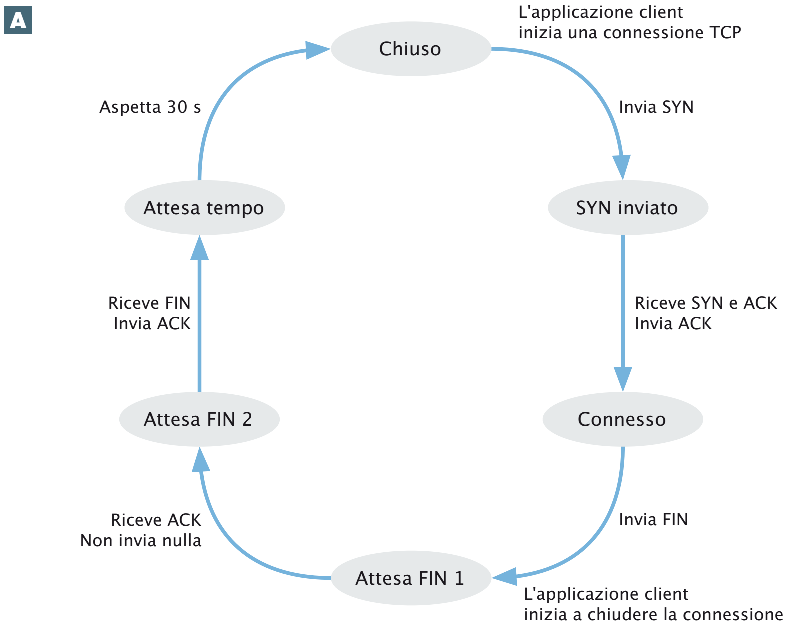
Al termine dello scambio di dati, entrambi gli host possono chiudere la connessione: quando questa termina, i buffer e le variabili vengono eliminati.

Supponiamo che il processo sul client decida di chiudere la connessione, emettendo un comando di chiusura a TCP e causando l'invio di uno speciale segmento al processo server. Questo segmento ha il bit FIN messo a 1: quando lo riceve, il server risponde con il proprio segmento di chiusura, con FIN posto a 1, al quale il client manda l'ACK finale.

A questo punto la connessione è interrotta.

Abbiamo parlato di apertura, gestione e chiusura di una connessione: durante la sua permanenza, il protocollo TCP che gira su ciascuno dei due host effettua transizioni fra diversi stati TCP.

Le figure 15A e 15B illustrano una tipica sequenza di stati, rispettivamente per il client e per il server.



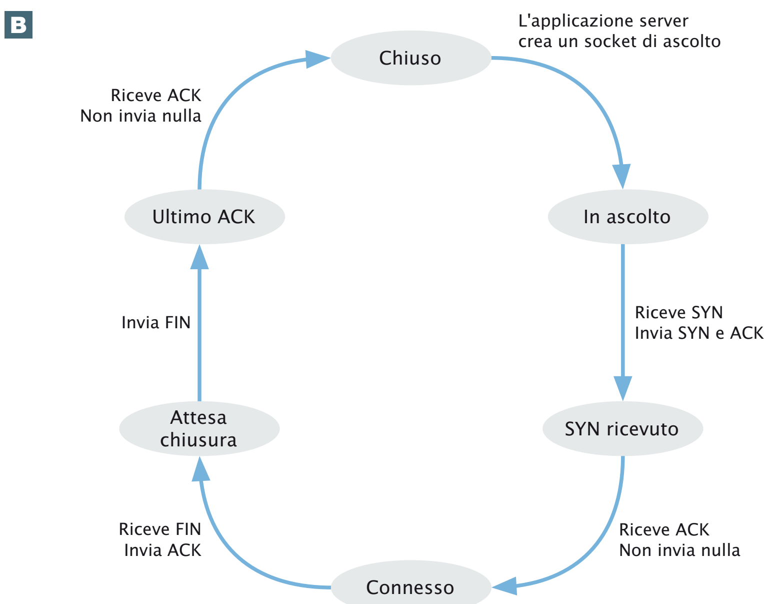


Figura **15**
Tipica sequenza di stati TCP:
(A) lato client
(B) lato server.

Indisponibilità della connessione • Sino a ora abbiamo ritenuto che entrambi gli host siano pronti a comunicare, ossia, come illustrato nella figura 15B, che il server sia in ascolto sulla porta cui il client invia il segmento SYN.

Vediamo, invece, cosa accade quando un host riceve un segmento TCP il cui numero di porta non corrisponde ad alcuno dei socket attivi.

Supponiamo, ad esempio, che un host riceva un segmento SYN con destinazione porta 80, ma che non accetti connessioni su quella porta (ciò significa che sull'host non sta girando un Web server sulla porta 80).

L'host trasmette allora uno speciale segmento di *reset* all'host mittente, con il bit RST posto a 1: in pratica, con questo segmento comunica al mittente che non ha un socket per il segmento ricevuto e, quindi, il segmento non deve essere ritrasmesso.

Concetti essenziali

- Un protocollo a livello trasporto fornisce una comunicazione logica tra due processi che girano su host diversi.
- I due principali protocolli usati da Internet sul livello di trasporto sono UDP e TCP.
- I protocolli a livello trasporto risiedono sugli host e non nei router.
- Nel multiplexing, il protocollo di invio a livello trasporto riceve blocchi provenienti da diverse applicazioni e li incapsula nei segmenti aggiungendo informazioni sulla provenienza e la destinazione dei blocchi.
- Nel demultiplexing, il protocollo di ricezione a livello trasporto riceve i pacchetti e li smembra, allocandoli alle corrette destinazioni in funzione delle informazioni relative ai blocchi.
- Il compito del protocollo UDP è limitato al multiplexing/ demultiplexing e alla verifica di eventuali errori.
- Diversamente da TCP, UDP non richiede la generazione di una connessione.
- La ridotta quantità di informazioni che UDP aggiunge al messaggio vero e proprio ne fa un protocollo "leggero", cioè che non determina ritardi significativi nella trasmissione.
- Il protocollo TCP può lavorare in modalità full-duplex: una volta stabilita la connessione, entrambi gli host possono ricevere e trasmettere tra loro contemporaneamente.
- TCP realizza un servizio di trasferimento affidabile dei dati attraverso due numeri, presenti nell'header, detti numero di sequenza e numero di riconoscimento.
- Il recupero di segmenti persi e la loro eventuale ritrasmissione si basano fondamentalmente sul timeout, scaduto il quale inizia la procedura di reinvio del pacchetto non ricevuto.
- Per evitare di saturare il buffer di ricezione, TCP fornisce un servizio di controllo di flusso, che consiste nell'adeguare la velocità di trasmissione alla velocità di ricezione all'altro capo della connessione.

Test

1 Dire se le seguenti affermazioni sono vere o false.

I seguenti sono protocolli usati nell'Internet a livello trasporto:

A PDU V F
B DPU V F
C UDP V F
D PPP V F

2 Dire se le seguenti affermazioni sono vere o false.

In qualsiasi host destinatario, il livello trasporto riceve i segmenti:

A dal sottostante livello rete V F
B dalla presa Ethernet V F
C dall'header attraverso un frame V F
D dai pacchetti demultiplati in fase di input V F

3 Dire se le seguenti affermazioni sono vere o false.

I protocolli a livello trasporto forniscono:

A un canale logico tra processi su client diversi V F
B un canale analogico tra processi su host diversi V F
C una comunicazione fisica tra processi su host diversi V F
D una comunicazione logica tra processi su host diversi V F

4 Dire se le seguenti affermazioni sono vere o false.

UDP e TCP sono:

A entrambi connectionless V F
B TCP connection oriented e UDP connectionless V F
C UDP connection oriented e TCP connectionless V F
D entrambi connection oriented V F

5 Dire se le seguenti affermazioni sono vere o false.

Rispetto a TCP, UDP è:

A molto più semplice V F
B molto più complesso V F
C fondamentalmente identico, solo opera con un checksum diverso V F
D molto più sofisticato nella ricostruzione di segnali degradati V F

6 Dire se le seguenti affermazioni sono vere o false.

I seguenti sono campi del segmento TCP:

A Numero di sequenza, Numero di intervallo V F
B Porta destinazione, Numero di riconoscimento V F
C Larghezza dell'header, Flag V F
D Checksum, Dati Urgenti V F

7 Dire se le seguenti affermazioni sono vere o false.

TCP è:

A full simplex V F
B half duplex V F
C multiplex V F
D full duplex V F

8 Dire se le seguenti affermazioni sono vere o false.

La gestione dei timeout permette di:

A evitare di trasmettere un segmento perduto V F
B ritrasmettere ogni segmento un numero dispari di volte V F
C evitare di ritrasmettere un segmento ancora in transito V F
D controllare l'esattezza del segmento ricevuto V F

9 Dire se le seguenti affermazioni sono vere o false.

TCP permette di:

A realizzare un trasferimento dati affidabile V F
B rendere inaffidabile il servizio affidabile del livello inferiore V F
C estrarre i segmenti in modalità interattiva V F
D eliminare la necessità della gestione dei timeout V F

10 Dire se le seguenti affermazioni sono vere o false.

TCP è in grado di effettuare:

A una rigenerazione dei segmenti perduti V F
B un controllo di flusso V F
C una gestione delle finestre di Windows V F
D l'eliminazione di tutti i timeout in condizioni di congestione V F

11 Dire se le seguenti affermazioni sono vere o false.

Con TCP, la variabile mantenuta dal destinatario per governare il flusso prende il nome di:

A porta di controllo del transito V F
B interrupt window V F
C soglia di saturazione V F
D finestra di ricezione V F

12 Dire se le seguenti affermazioni sono vere o false.

Se un host riceve un segmento TCP il cui numero di porta non corrisponde nessuno dei socket attivi, l'host:

A attende passivamente che il mittente invii un segmento con un numero d porta idoneo V F
B genera un nuovo socket attivo e avvia la ricezione V F
C invia un messaggio di reset all'host trasmittente V F
D corregge il segmento ponendo il numero di porta corrispondente a un socket attivo V F

4 Il livello rete

Il capitolo è focalizzato sull'analisi del livello rete, cioè di quella parte della struttura del network cui è demandato il compito di prendere in consegna i blocchi di dati del livello superiore, inscriverli in datagram e farli pervenire all'host destinatario.

Dietro un'azione descritta in poche parole si celano una notevole complessità e più soluzioni possibili a medesimi problemi o sottoproblemi: attraverso esempi tratti dall'esperienza quotidiana, l'allievo viene posto a confronto con aspetti della gestione del traffico dati la cui soluzione ha permesso, negli ultimi decenni, la nascita e il successo di Internet, la rete delle reti.

La trattazione dell'argomento, molto tecnica e di alto livello, pone comunque in evidenza quali e quanti diversi temi debbano essere presi in considerazione da parte di coloro che offrono servizi di comunicazione dati di qualità.

1 Inoltro e instradamento

Abbiamo visto nel capitolo 1 che il livello rete provvede a prendere i segmenti consegnati dal livello superiore, inserirli in altri blocchi chiamati **datagram**, e fornire il servizio di consegna all'indirizzo richiesto dal livello trasporto. Il ruolo del livello rete, dunque, è apparentemente semplice: muovere i pacchetti dall'host mittente all'host destinatario. Per fare ciò, sono necessarie due importanti funzioni: **inoltro** (*forwarding*) e **instradamento** (*routing*):

- *inoltro*: quando un pacchetto arriva a uno degli ingressi di un router, questo deve veicolarlo all'uscita corretta. Ad esempio, facendo riferimento alla figura 1 (pagina a lato), un pacchetto che arriva dall'host H1 al router R1, deve essere spedito da questo al router successivo, su un percorso che lo porti fino all'host destinatario H2
- *instradamento*: il livello rete deve definire il percorso che il pacchetto compie nel passaggio dall'host mittente a quello destinatario, ossia a quale router successivo il pacchetto deve essere spedito. Sempre in riferimento alla figura 1, si vede che il router R1 indirizza a R2, che a sua volta indirizza a R3, il quale invia a R4, per finire a R5, cui è connesso il destinatario H2. In questo caso, il percorso è praticamente obbligato, ma nella realtà sono spesso disponibili molti percorsi alternativi. Per il calcolo del percorso, in questi casi vengono utilizzati particolari algoritmi di routing, di cui accenneremo più avanti

I termini *inoltro* e *instradamento* sono spesso usati indifferentemente per descrivere l'intero servizio del livello rete ma, se vogliamo definirli correttamente, il primo (*inoltro*) si riferisce all'azione, interna al router, di trasferimento del pacchetto da un ingresso a una determinata uscita; il secondo (*instradamento*) indica il processo, che coinvolge l'intera rete, di definizione del percorso su cui il pacchetto viene instradato per giungere dal mittente al destinatario. Utilizzando una analogia, potremmo considerare l'*inoltro* come la scelta dell'uscita da prendere a uno svincolo per dirigerci verso la nostra destinazione, e l'*instradamento* come la definizione di quale, tra i vari percorsi possibili, utilizzeremo per andare da un luogo a un altro.

Tabelle di inoltro • All'interno dei router si trovano le cosiddette **tabelle di inoltro**: il router inoltra un pacchetto esaminando il valore contenuto in un determinato campo

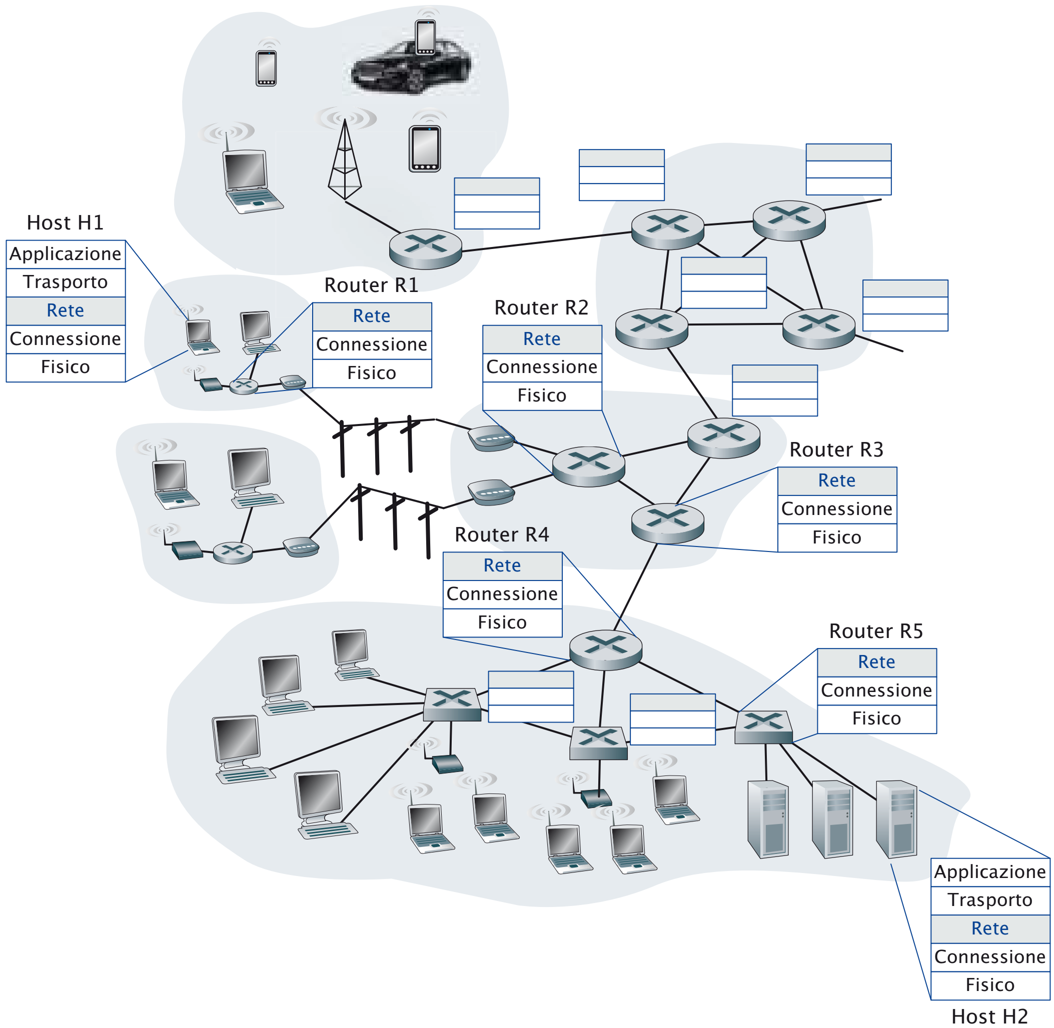


Figura **1**
Il livello rete all'interno dell'Internet.

dell'header e confrontandolo con quelli contenuti nella tabella, ai quali corrisponde una e una sola uscita possibile. Notiamo come non sia vero il contrario, in quanto a una uscita possono corrispondere molti valori del campo.

La configurazione delle tabelle di inoltro è definita dagli algoritmi di routing. Questi possono essere *centralizzati*, ossia eseguiti in un sito centrale e poi scaricati sui singoli router, oppure *distribuiti*, una parte dei quali autonomamente all'interno dei router. Sia che si tratti di algoritmi centralizzati che nel caso di algoritmi distribuiti, ogni router coinvolto, in ogni caso, dovrà configurare la sua tabella di inoltro, per permettere il passaggio del pacchetto al router successivo, attraverso messaggi definiti dal protocollo adottato.

La letteratura tecnica dei vari produttori ha causato molta confusione nella terminologia riguardante i router: ad esempio, i router che possiedono interfacce Ethernet vengono frequentemente chiamati "*switch di livello 3*". Per correttezza, invece, dovremmo chiamare *switch* solo gli apparati che basano le loro decisioni di inoltro sui valori contenuti nei frame, ossia che operano al livello connessione: in realtà, qualunque router ha implementato al suo interno anche la gestione del livello 2, altrimenti non potrebbe fornire i servizi del livello 3.

In alcune architetture di rete, ad esempio **ATM** (*Asynchronous Transfer Mode*), **frame relay** e **MPLS** (*Multi Protocol Label Switching*), è presente un'ulteriore importante funzione di livello 3, chiamata **connection setup** (*configurazione della connessione*). Queste architetture, infatti, richiedono che i router lungo il percorso, prima che i pacchetti possano fluire su di esso, impostino il proprio stato eseguendo uno handshake l'uno con l'altro.

Modelli di servizio • Esaminiamo i differenti tipi di servizi che possono essere forniti dal livello rete ai livelli superiori. Per far questo, utilizzeremo il modello di servizio che definisce le caratteristiche del trasporto dei pacchetti tra il mittente e il destinatario.

Nell'host mittente, quando il livello trasporto passa un pacchetto al livello rete, questo può fornire i seguenti servizi:

- *consegna garantita*: garantisce che il pacchetto giungerà a destinazione
- *consegna garantita con ritardo limite definito*: garantisce che il pacchetto giungerà a destinazione con un ritardo inferiore a un tempo definito

Inoltre, a un flusso di pacchetti tra mittente e destinatario possono essere forniti i seguenti servizi:

- *consegna ordinata*: garantisce che i pacchetti giungano a destinazione nella stessa sequenza con cui sono stati inviati
- *banda minima garantita*: emula il comportamento di una *connessione a bit rate costante* tra i due host: finché il mittente invia i pacchetti a una velocità inferiore al bit rate specificato, non si hanno perdite di pacchetti e ogni pacchetto arriva con un ritardo totale predefinito
- *jitter massimo garantito*: la limitazione del **jitter** garantisce che l'intervallo di trasmissione tra due pacchetti generato dal mittente sia uguale - o diverso per un valore specificato - all'intervallo con cui vengono ricevuti dal destinatario
- *servizi di sicurezza*: usando una chiave segreta di sessione conosciuta solo dai due host, il livello rete può crittografare i dati contenuti nei datagram al momento dell'invio da parte del mittente. Lo stesso livello rete provvede anche a decrittare i dati presso il destinatario. In questo modo, viene garantita la segretezza a tutti i segmenti del livello trasporto tra i due host. In aggiunta, il livello trasporto può fornire servizi di autenticazione della fonte

Quelli sopra elencati sono solo i principali servizi che il livello rete è in grado di fornire. Tuttavia, in molti casi, nessuno di essi è effettivamente implementato: l'Internet, ad esempio, fornisce un singolo servizio, chiamato **best-effort** (termine intraducibile che significa "*al meglio delle possibilità istantanee della rete*"). Questo modello di servizio non garantisce praticamente nulla, ma esistono precise ragioni, che vedremo più avanti, per cui viene utilizzato nell'Internet. Esistono, comunque, architetture di rete che offrono modelli di servizio che vanno molto oltre il *best-effort* dell'Internet: le reti ATM, per esempio, forniscono modelli di servizio multipli, ossia possono gestire, all'interno della stessa rete, connessioni con diverse classi di servizio.

Abbiamo visto nel capitolo precedente che il livello trasporto può fornire un servizio UDP (*connectionless*) o TCP (*connection-oriented*) tra due applicazioni. Allo stesso modo, il livello 3 può fornire un servizio con oppure senza connessione tra due host, anche se tra i servizi del livello 2 e quelli del livello 3 sussistono alcune differenze fondamentali:

- i servizi a livello rete fra host e host sono forniti al livello trasporto, mentre i servizi del livello trasporto sono servizi tra due processi che vengono forniti al livello applicazione
- in tutte le principali architetture di rete attuali, il livello 3 fornisce un servizio connectionless o un servizio connection-oriented, ma non entrambi. Le reti che usano il servizio connectionless sono dette **datagram network**, quelle che usano il servizio connection-oriented sono dette **Virtual Circuit Network** (*reti a circuito virtuale)* o **reti VC**
- l'implementazione del servizio connection-oriented è fondamentalmente diversa nei due livelli. Abbiamo visto che a livello trasporto il servizio è localizzato al bordo della rete, all'interno dell'host: a livello rete, invece, è localizzato sia negli host che dentro ai router del nucleo di rete

Reti a circuito virtuale • Un **VC** (circuito virtuale) consiste in un percorso - ossia una sequenza di tratte e di router - tra mittente e destinatario, con un numero VC assegnato per ogni tratta lungo il percorso, e valori nelle tabelle di inoltro dei router.

Un pacchetto appartenente a un circuito virtuale contiene un numero VC nel proprio header; poiché un circuito virtuale può avere differenti numeri VC nelle varie tratte, ogni router sul percorso deve rimpiazzare il contenuto dell'header di ogni pacchetto con un nuovo numero, che ricava dalla corrispondente tabella di inoltro.

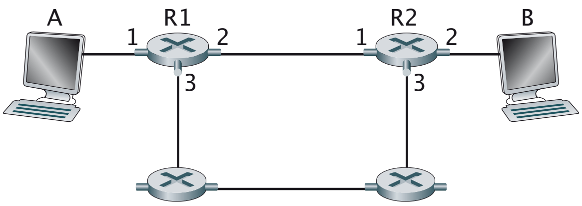


Figura **2**
Un semplice circuito virtuale.

Consideriamo la piccola rete di figura 2. Se l'host A richiede l'attivazione di un circuito virtuale con l'host B, la rete stabilisce il percorso A-R1-R2-B e assegna nella tabella di inoltro tre numeri, ad esempio 12, 22 e 32 rispettivamente alle tratte A - R1, R1 - R2, R2 - B.

Quando un pacchetto lascia l'host A, il suo header contiene il numero VC 12, quando lascia il router R1 contiene 22 e quando lascia R2 contiene 32. Ciò avviene perché la tabella di inoltro di ogni router contiene i numeri relativi a ogni tratta entrante e uscente del circuito virtuale: ogni volta che viene generato un nuovo VC, un nuovo valore è aggiunto alla tabella e, quando il circuito viene terminato, i relativi valori sono rimossi dalla tabella.

La ragione per cui i numeri VC cambiano da una tratta all'altra è molto semplice: in questo modo non è necessario che tutti i router sul percorso siano costretti a negoziare un unico numero per ogni circuito. In questo modo si semplifica molto il processo di generazione del circuito stesso: ogni router, infatti, è connesso normalmente a un numero elevato di altri nodi, ognuno dei quali può avere a sua volta molti circuiti virtuali attivi con altri router. La propagazione e la negoziazione tra tutti i router di un unico numero VC potrebbero richiedere un tempo piuttosto elevato, se non diventare impossibile a causa di un eccessivo numero di nodi.

Potendo definire un numero per ogni tratta, la negoziazione è limitata a due nodi, quindi molto più semplice e veloce.

Da quanto detto, si capisce come in una rete VC ogni router deve mantenere una registrazione dello stato di connessione per i circuiti attivi in ogni momento.

Nell'attività di un circuito virtuale possiamo identificare tre fasi temporali:

- *setup*: durante questa fase, il livello trasporto del mittente contatta il livello rete, specifica l'indirizzo del destinatario e attende la generazione del circuito virtuale. Il livello rete determina il percorso tra i due estremi, ossia la serie di tratte e di nodi attraverso cui viaggeranno i pacchetti. Inoltre, definisce i numeri VC per ogni singola tratta e li aggiunge alle tabelle di inoltro dei router lungo il percorso. In questa fase, il livello rete può anche definire le risorse riservate allo specifico circuito, ossia definire i livelli di servizio forniti
- *trasferimento dati*: una volta che il circuito virtuale è stabilito, i pacchetti iniziano a fluire lungo il circuito virtuale
- *teardown* (o *smantellamento*): il processo inizia quando uno dei due host informa il livello rete che desidera chiudere il circuito virtuale. Il livello rete informa l'altro host della fine della connessione e aggiorna le tabelle di inoltro di tutti i router interessati per indicare che il circuito virtuale non esiste più

È importante notare la differenza tra il setup del circuito virtuale a livello 3 e il setup della connessione a livello 4: il setup a livello trasporto coinvolge solo i due host, mentre i nodi della rete non ne hanno alcuna percezione; il setup a livello rete coinvolge sia i due host sia tutti i router sul percorso, e ogni router è a conoscenza di tutti i VC che passano attraverso di esso.

I messaggi che passano tra i due host per iniziare o terminare un VC e quelli che i router si scambiano per modificare lo stato delle connessioni nelle tabelle di routing vengono detti **messaggi di segnalazione** (*signaling messages*) e i protocolli utilizzati per lo scambio **protocolli di segnalazione** (*signaling protocols*).

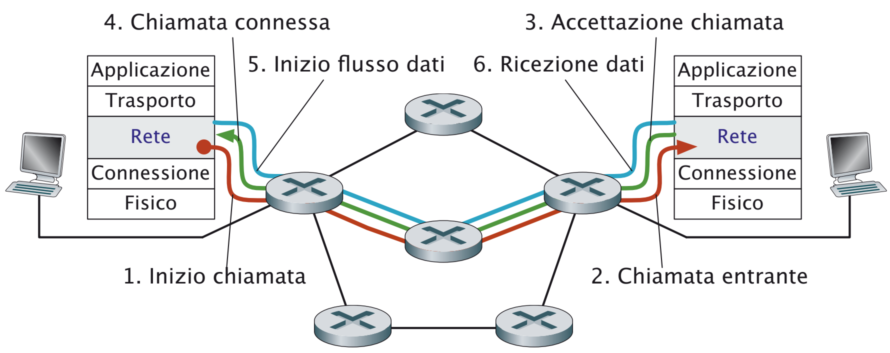


Figura **3**
Setup di un circuito virtuale.

Reti datagram • In una rete datagram, ogni volta che un host vuole inviare un pacchetto, lo marca con l'indirizzo del destinatario e lo spinge nella rete: non esiste alcun setup, e i router non mantengono alcuna informazione di stato. Ogni router confronta l'indirizzo di destinazione del pacchetto con la propria tabella di inoltro, che mappa gli indirizzi di destinazione sulle proprie interfacce in uscita, per poi inoltrarlo sulla tratta successiva.

Supponiamo che tutti gli indirizzi di destinazione siano di 32 bit - che è proprio la lunghezza di un indirizzo IP - e, quindi, esistano 4 294 967 296 possibili destinatari: è chiaro che la tabella all'interno del router non può contenere un simile numero di valori. È necessario, quindi, accorpare al massimo gli indirizzi.

Se, ad esempio, il router ha quattro interfacce, numerate da 0 a 3, e i pacchetti vanno instradati per gruppi, possiamo trovare questa situazione:

La tabella di inoltro diventa ora estremamente semplice, presentando solo quattro valori:

Gruppo di indirizzi del destinatario	Interfaccia
11001000 00010111 00010000 00000000 sino a 11001000 00010111 00010111 11111111	**0**
11001000 00010111 00011000 00000000 sino a 11001000 00010111 00011000 11111111	**1**
11001000 00010111 00011001 00000000 sino a 11001000 00010111 00011111 11111111	**2**
altrimenti	**3**

In questo modo, il router accoppia alle interfacce di uscita solo un prefisso (la parte iniziale) dell'indirizzo di destinazione. Se, ad esempio, l'indirizzo è 11001000 00010111 00010110 10100001, i primi 21 bit, che costituiscono il cosiddetto prefisso, coincidono con il primo valore della tabella dei prefissi, quindi il router indirizza il pacchetto sull'uscita 0, indipendentemente dai bit successivi.

Prefisso	Interfaccia
11001000 00010111 00010	**0**
11001000 00010111 00011000	**1**
11001000 00010111 00011	**2**
altrimenti	**3**

Se il prefisso non corrisponde ad alcun valore in tabella, il router indirizza il pacchetto sull'uscita 3. Quando esistono corrispondenze multiple, come può avvenire in tabelle più complesse, il router usa la regola del prefisso più lungo, ossia inoltra il pacchetto verso l'uscita che ha la corrispondenza con il maggior numero di bit.

In una rete datagram, le tabelle di inoltro vengono modificate dagli algoritmi di routing, che le aggiornano mediante ogni pochi minuti. Poiché queste modifiche possono avvenire in qualunque momento, una serie consecutiva di pacchetti inviati da un host all'altro può seguire più di un percorso attraverso la rete e, quindi, giungere al destinatario in modo disordinato.

A4-01
I due tipi di rete

2 Funzione di inoltro

Dopo aver esaminato le funzioni e i servizi del livello rete, entriamo più in profondità sulla funzione di inoltro, di cui abbiamo appena visto alcune caratteristiche, quali le tabelle di inoltro e l'indirizzamento. Nel gergo delle reti, i termini inoltro (*forwarding*) e commutazione (*switching*) sono spesso utilizzati come sinonimi, anche se la seconda dovrebbe essere riservata alle funzioni di livello 2. Nel seguito, ci adegueremo a quest'ultima convenzione.

Router • L'architettura generale di un router è mostrata in figura 4.

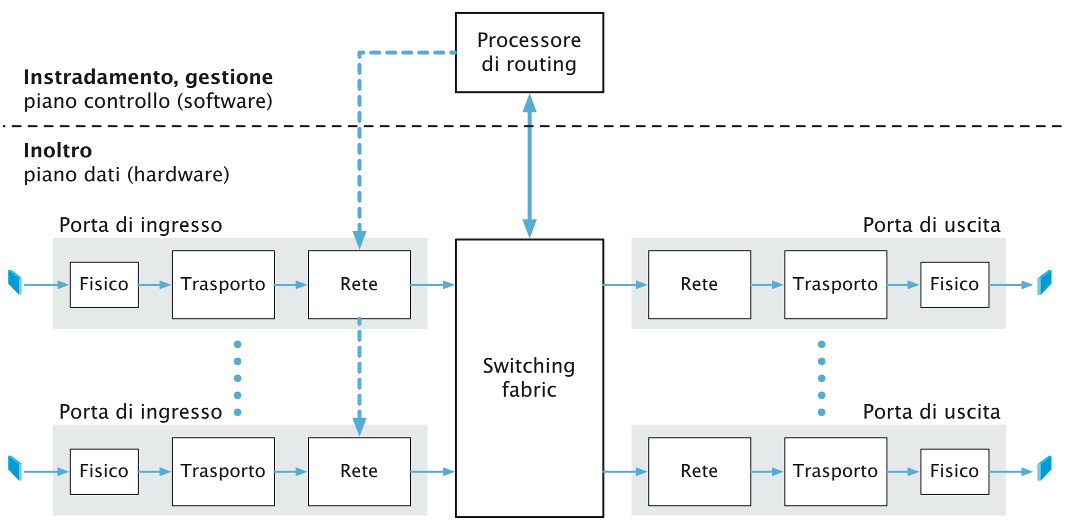


Figura **4**
Architettura generale di un router.

Come si vede, possiamo identificare quattro componenti base:

- *porte di ingresso*: una porta di ingresso svolge più funzioni fondamentali, a diversi livelli. A livello fisico, provvede a terminare la tratta fisica in ingresso; a livello connessione, gestisce le funzioni necessarie per interoperare con il livello link dall'altro lato della tratta; a livello rete, effettua la ricerca nella tabella di inoltro per definire a quale uscita inviare i pacchetti entranti. Notiamo come il termine "*porta*", che si riferisce all'interfaccia fisica di ingresso e uscita del router, abbia un significato distinto e diverso rispetto alle porte logiche associate alle applicazioni di rete e con il socket
- *switching fabric* (letteralmente *tessuto di commutazione*, viene sempre nominato in inglese): serve a connettere le porte di ingresso del router alle porte di uscita: si tratta, in pratica,

di una matrice di commutazione che funziona come una rete dentro la rete, instradando i pacchetti da una porta all'altra

- *porte di uscita*: una porta di uscita accumula i pacchetti ricevuti dallo switching fabric e li trasmette alla tratta in uscita, operando, come le porte di ingresso, anche a livello fisico e a livello connessione. Quando la connessione è bidirezionale, le relative porte di ingresso e uscita sono accoppiate in un unico connettore
- *processore di routing*: si tratta di una particolare CPU, che esegue i protocolli di routing, mantiene le tabelle di routing e le relative informazioni sullo stato delle connessioni, e calcola le tavole di inoltro. Inoltre, gestisce le funzioni di network management, cui accenneremo più avanti

Le porte di ingresso e uscita e lo switching fabric, che a volte sono chiamati nel loro insieme "*piano di inoltro del router*", sono realizzati via hardware per ragioni di velocità: consideriamo che, in una connessione a 10 Gbps, se una porta di ingresso riceve un datagram di 64 byte, ha solo 51,2 ns per processarlo, prima che arrivi il successivo. Se *N* porte sono combinate in un'unica scheda, il sistema di processo dei datagram deve operare *N* volte più velocemente: si tratta di prestazioni impossibili da ottenere via software. Le funzioni di controllo del router, invece, operano nella scala dei millisecondi e, quindi possono essere gestite via software.

Per capire meglio il funzionamento di un router, possiamo paragonarlo a una rotonda cui fanno capo molte strade, tutte a senso unico, alcune in entrata e altre in uscita, con un casello a ogni entrata. Ogni auto che arriva deve comunicare al casellante la propria destinazione finale, e questi indica quale uscita l'auto prendere e ne apre la sbarra. Arrivata all'uscita, l'auto può trovarne altre che si stanno infilando nella stessa strada, e quindi deve accodarsi.

Usando questa analogia, vediamo chiaramente quali situazioni possono dar luogo a colli di bottiglia:

1) l'auto arriva molto veloce, ma il casellante è lento
2) il casellante è veloce, quindi non genera code, ma la rotonda è già piena di auto e il traffico è quindi molto lento, malgrado siano disponibili più corsie
3) molte auto vogliono lasciare contemporaneamente la rotonda dalla stessa uscita
4) è necessario dare priorità a determinati tipi di veicoli
5) è necessario impedire a determinati tipi di veicoli di entrare nella rotonda per primi

Elaborazione di ingresso e uscita • Abbiamo visto che le porte di ingresso effettuano la ricerca nella tabella di inoltro, per determinare a quale porta di uscita inviare i pacchetti. La tabella è calcolata e aggiornata dal processore di routing, ma normalmente ne esiste una copia locale in corrispondenza di ogni singola porta: in questo modo, le decisioni di inoltro possono essere prese localmente, evitando di invocare il processore di routing per ogni pacchetto e, quindi, eliminando il rischio di un collo di bottiglia. Poiché, comunque, la ricerca deve essere compiuta entro tempi estremamente ridotti, la semplice ricerca lineare è troppo lenta, e vengono invece utilizzati particolari algoritmi di ricerca veloce.

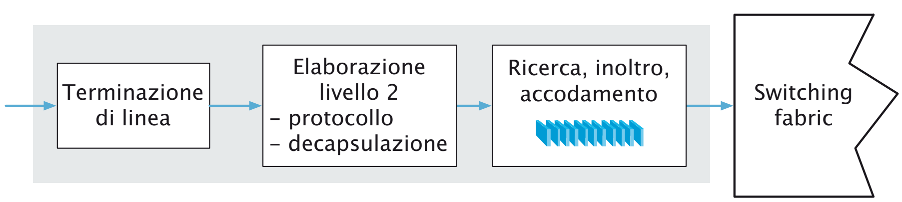


Figura **5**
Elaborazione all'interno di una porta di ingresso.

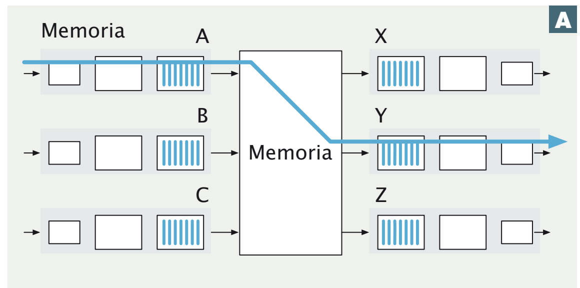
Una volta che è stata determinata la porta di uscita, il pacchetto può essere inviato nello switching fabric. In alcuni tipi di router un pacchetto può essere temporaneamente bloccato, se altri pacchetti stanno usando lo switching fabric: il pacchetto bloccato viene accodato alla porta di ingresso e programmato per l'inserimento in un momento successivo. Vedremo più avanti in maggior dettaglio il processo di blocco, accodamento e temporizzazione.

Oltre a provvedere alla ricerca nella tabella di inoltro, la porta di ingresso deve controllare e aggiornare alcuni campi presenti nell'header del pacchetto e deve far avanzare i contatori utilizzati per l'amministrazione di rete.

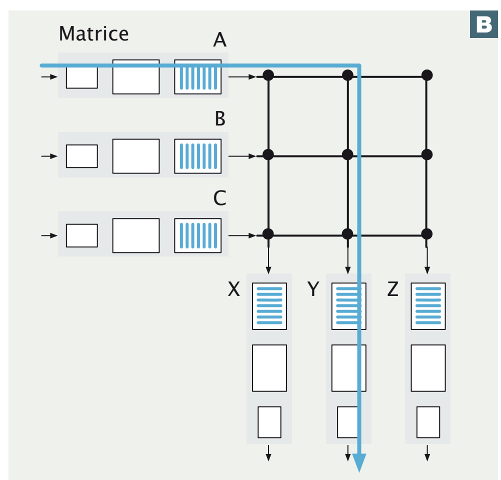
L'elaborazione in uscita è molto più semplice, dato che la porta si limita a selezionare i pacchetti che si trovano nella memoria di uscita e disaccodarli per la trasmissione.

Commutazione • Lo switching fabric è il vero cuore del router, dato che i pacchetti fluiscono attraverso di esso per andare dalla porta di ingresso a quella di uscita. Esistono diverse strutture dello switching fabric, che provvedono a veicolare i pacchetti in modo diverso: mediante una **memoria** (figura 6A), attraverso una **matrice** (figura 6B) oppure attraverso un **bus** (figura 6C).

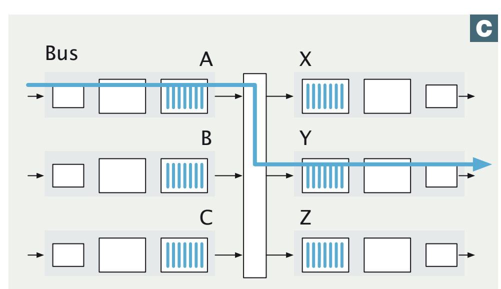
Figura **6**
Tipologie di switching fabric in un router: [A] commutazione via memoria, [B] via bus e [C] via matrice.



- *commutazione via memoria*: i primi, semplici router erano computer tradizionali, che effettuavano l'inoltro direttamente tramite la CPU, utilizzando la memoria RAM. Le porte funzionavano come I/O tradizionali in un normale sistema operativo e il processore veniva avvisato della presenta di un pacchetto entrante da un *interrupt*. A questo punto, il processore copiava il pacchetto in memoria, ne estraeva l'indirizzo di destinazione, trovava la porta di uscita e copiava il pacchetto dalla memoria al buffer della porta scelta. Ovviamente, si trattava di un processo molto lento, limitato dalla velocità di lettura e di scrittura della memoria.
  Molti router moderni commutano attraverso la memoria: la differenza fondamentale con i router di prima generazione sta nel fatto che la ricerca nella tabella di inoltro e la registrazione nella cella di memoria appropriata vengono effettuate direttamente dalla porta di ingresso. In ogni caso, si tratta di un'architettura usata in router che non devono trattare grandi quantità di dati.



- *Commutazione via bus*: in questa architettura, la porta di ingresso trasferisce direttamente i pacchetti alla porta di uscita attraverso un bus condiviso, senza intervento da parte del processore di routing. La porta di ingresso appende al pacchetto un'etichetta che indica la porta di uscita cui deve essere trasferito e lo invia sul bus: tutte le porte di uscita ricevono il pacchetto, ma solo la porta che corrisponde all'etichetta lo trattiene. Prima di instradare il pacchetto, la porta di uscita elimina l'etichetta, che serve solo all'interno del router.
  Poiché il bus può trasportare solo un pacchetto per volta, se più pacchetti arrivano contemporaneamente su porte diverse, tutti tranne uno devono aspettare: tornando al paragone della rotonda, è come se nella rotonda potesse circolare solo un'auto per volta. Poiché ogni pacchetto deve attraversare l'unico bus disponibile, la velocità di commutazione del router è limitata dalla velocità del bus. Dato che le tecnologie attuali permettono di raggiungere bande passanti di decine di Gbps, questa architettura si rivela adatta a reti locali, anche piuttosto complesse e performanti.



- *Commutazione via matrice*: per superare le limitazioni di banda di un singolo bus condiviso, è possibile utilizzare una struttura a matrice, dotata di $2N$ bus, che connettono N porte di ingresso con N porte di uscita. Ogni nodo tra i bus consiste in uno switch (interruttore), che può essere aperto e chiuso dalla logica interna dello switching fabric.
  Facendo riferimento alla figura 6C, quando un pacchetto arriva alla porta A e deve essere inviato alla porta Y, il controllore dello switching fabric chiude la matrice all'intersezione tra il bus A e il

bus Y. A differenza delle altre architetture, è possibile inoltrare più pacchetti contemporaneamente, purché siano usati bus di ingresso e uscita liberi: ad esempio la porta B può inviare alla porta X senza influenzare l'altra connessione. Se, però, due porte di ingresso cercano di inoltrare due pacchetti alla stessa porta di uscita, uno di essi deve attendere all'ingresso, dato che ogni bus non può che trasportare un solo pacchetto alla volta.
Esistono architetture delle matrici più sofisticate, che prevedono diversi stadi di commutazione, in modo da poter far giungere contemporaneamente più pacchetti sulla stessa porta di uscita.

Colli di bottiglia • Abbiamo visto come gli accodamenti possano formarsi sia sulle porte di ingresso che di uscita: la posizione e la dimensione delle code dipendono dal carico di traffico, dalla velocità dello switching fabric e dalla velocità delle linee in entrata e in uscita. Se le code crescono eccessivamente, la memoria del router può esaurirsi, causando il fenomeno della perdita di pacchetti di cui abbiamo parlato nel capitolo 1.

Supponiamo per semplicità che le linee in ingresso e in uscita abbiano una uguale velocità di trasmissione, in pacchetti al secondo, che indichiamo con R_{linea}. Supponiamo anche che il router abbia N ingressi e N uscite, che i pacchetti abbiano tutti la stessa lunghezza e che arrivino alle porte di ingresso in modo sincrono. Se la velocità di trasferimento R_{switch} dello switching fabric è N volte più veloce di R_{linea}, allora l'accodamento alle porte di ingresso sarà di entità trascurabile: anche nel caso peggiore, in cui a tutte le N porte ricevono un pacchetto contemporaneamente e tutti i pacchetti sono inoltrati sulla stessa porta di uscita, la velocità di smaltimento dello switching fabric è tale da prelevare tutti i pacchetti prima che ne arrivi un secondo gruppo.

Alle porte di uscita, però, la situazione cambia completamente. Se la velocità delle linee di uscita è la stessa delle linee in entrata - cosa molto probabile - e tutti i pacchetti in entrata arrivano alla stessa porta di uscita, questa dovrà accodarli, dato che non può inviarne più di uno alla volta: la coda, quindi crescerà di $(N–1)$ pacchetti a ogni gruppo che arriva, saturandosi velocemente. In questo caso alcuni pacchetti vengono scartati e, quindi, persi.

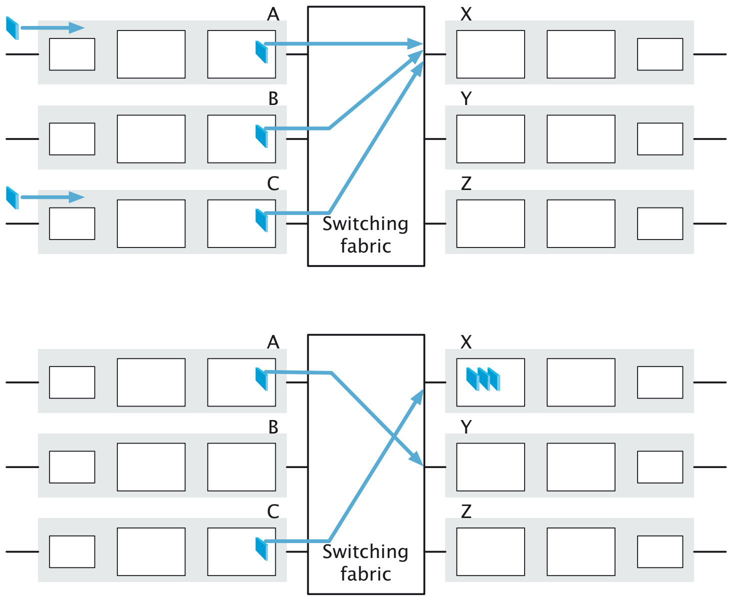


Figura **7**
Accodamento di pacchetti sulla porta di uscita.

Nella realtà questa situazione teorica non si presenta mai, ma sono comunque necessari buffer di grandi dimensioni: la formula che si utilizza per calcolare la capacità del buffer deriva sia da studi teorici, sia da osservazioni pratiche. I pacchetti sono trasportati da un flusso TCP che presenta, su un determinato link di capacità C, un tempo medio di andata e ritorno RTT ben definito (il tempo di andata e ritorno definisce gli intervalli di tempo in cui i pacchetti si presentano all'ingresso). Se è presente un numero N di flussi TCP, con N grande, allora la dimensione del buffer è:

$$B = RTT \cdot \frac{C}{\sqrt{N}}$$

Se, ad esempio, consideriamo un link a 10 Gbps, un RTT del flusso TCP di 250 ms e 100 flussi TCP, allora il buffer dovrà avere una capacità:

$$B = 0,25 \cdot \frac{10}{\sqrt{100}} = 0,25 \text{ Gb} = 250 \text{ Mb}$$

Conseguenza del fenomeno di accodamento sulle porte di uscita è la necessità di definire un criterio in base al quale scegliere il pacchetto, tra tutti quelli presenti in coda, che verrà trasmesso per primo. Si può andare da una regola molto semplice, ad esempio "*primo entrato, primo uscito*" o **FCFS** (*First Come First Served*), fino a strategie sofisticate, del tipo di quelle che distribuiscono la scelta di pacchetti da trasmettere tra tutti i flussi che hanno pacchetti accodati, ad esempio la **WFQ**, (*Weighted Fair Queueing*).

Allo stesso modo, se non vi è posto sufficiente nel buffer di ingresso, è necessario decidere se il pacchetto entrante va scartato, o se vanno rimossi altri pacchetti all'interno del buffer per far posto al nuovo arrivato.

In alcuni casi, può essere vantaggioso scartare un pacchetto prima che la coda sia piena, in modo da mandare un segnale di "*congestione in vista*" al mittente. La scelta tra queste possibilità viene fatta utilizzando particolari algoritmi detti di "*gestione attiva delle code*" o **AQM** (*Active Queue Management*).

Per finire, vediamo come anche lo stesso switching fabric può generare code sulla porta di ingresso.

Se due pacchetti provenienti da porte di ingresso diverse sono diretti alla stessa porta di uscita, uno dei due dovrà attendere che l'altro sia accettato nel buffer della porta di uscita.

Facendo riferimento alla figura 8, ammettiamo che sia la porta C a dover attendere l'inoltro verso la porta X.

Se la porta C ha accodati altri pacchetti, questi non potranno essere inoltrati neppure se sono diretti a un'altra porta di uscita, ad esempio la Y: il pacchetto in attesa della porta X, infatti, blocca l'uscita della porta C. Questo fenomeno è conosciuto col nome di "*blocco in cima alla fila*" o **HOL** (*Head Of the Line blocking*).

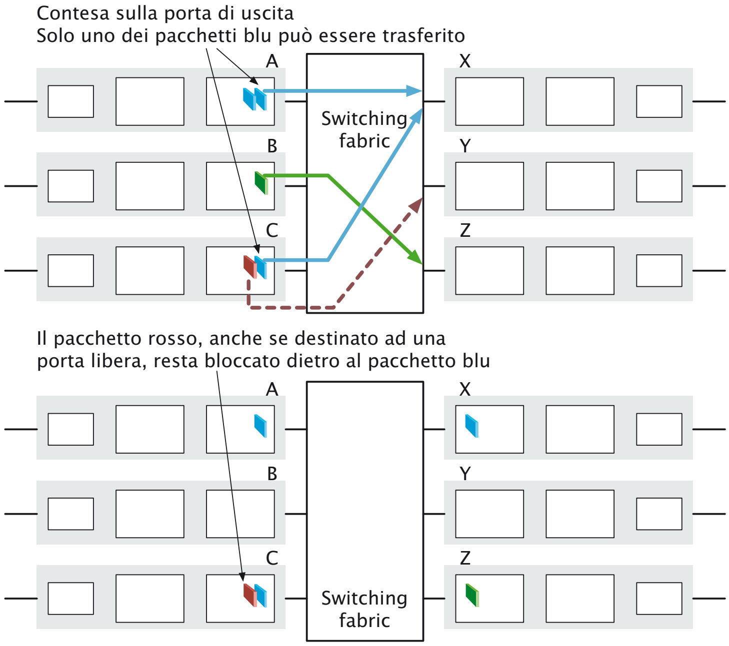


Figura **8**
Blocco di un pacchetto in cima alla fila.

3 Protocolli

Sinora abbiamo parlato di inoltro, instradamento e indirizzamento al livello rete senza fare riferimento ad alcuna specifica rete di computer. Da qui in avanti, vedremo come il livello rete è configurato sull'Internet, tenendo conto che anche la maggior parte delle LAN utilizza oggi gli stessi protocolli.

Il livello rete dell'Internet presenta tre componenti principali:

- *protocollo IP*: è il protocollo che provvede all'indirizzamento, quindi all'instradamento, al livello rete, garantendo l'interoperabilità tra reti che utilizzano strutture diverse al livello inferiore. Come abbiamo visto, si tratta di un protocollo senza connessione, che fornisce solo un servizio best-effort: quindi non garantisce alcun controllo di errore, di flusso e di congestione, controlli che sono demandati ai protocolli del livello superiore
- *protocolli di routing*: determinano il percorso che i datagram seguiranno dal mittente al destinatario e calcolano le tabelle di inoltro
- *protocollo ICMP*: si tratta di un protocollo che riporta gli errori nei datagram e una serie di informazioni riguardanti il livello rete.

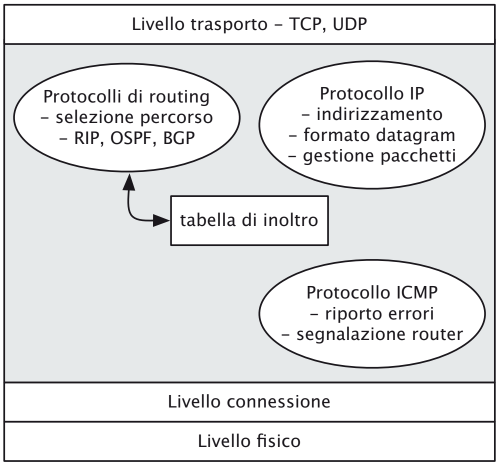


Figura **9**
Componenti principali del livello rete.

Protocollo IP • Esistono due diverse versioni del protocollo IP: **IPv4**, cioè la versione 4 del protocollo IP, che è quella utilizzata sino a oggi, e **IPv6** (la versione 6), che è destinata a sostituire la 4 nei prossimi anni, con un processo graduale che è iniziato nel 2012.

Vediamo innanzitutto il formato del datagram utilizzato dall'IPv4 (vedi anche figura 10):

- *numero di versione*: dichiara la versione del protocollo IP che si sta usando. Leggendolo, il router determina come interpretare il resto del datagram
- *lunghezza dell'header*: poiché l'header può contenere un numero variabile di opzioni, questo campo è necessario per definire dove iniziano i dati all'interno del datagram
- *tipo di servizio* **TOS** (*Type Of Service*): questo campo permette di diversificare i datagram, in modo da permettere ai router di fornire diversi livelli di servizio in funzione dell'informazione trasportata: le necessità dello streaming video o del VoIP, ad esempio, sono ben diverse da quelle di un trasferimento FTP
- *lunghezza del datagram*: indica la dimensione totale del datagram. Poiché il campo ha una lunghezza di 16 bit, la dimensione totale potrebbe arrivare a 65.536 byte (anche se difficilmente un datagram supera i 1.500 byte)
- *identificatore, flag* e *offset di frammentazione*: forniscono indicazioni ai router riguardanti la frammentazione dei datagram che, come vedremo più avanti, può essere necessaria per poterli incapsulare nei frame del livello inferiore
- *tempo di vita* **TTL** (*Time To Live*): indica il tempo massimo di permanenza del datagram in rete, per evitare che circoli all'infinito nel caso in cui non venga instradato correttamente. A ogni inoltro, il valore del campo viene decrementato di un'unità: quando arriva a 0, il datagram viene eliminato
- *protocollo superiore*: quando il datagram raggiunge il destinatario, questo campo permette di decidere quale protocollo di livello trasporto deve prendere in carico i dati in esso contenuti. Ad esempio, se il valore del campo è 6 i dati sono passati a TCP, mentre se è 17 i dati sono passati a UDP. Il numero di protocollo nel datagram IP ha lo stesso ruolo del numero

di porta nel segmento del livello trasporto: il numero di protocollo è il collante che unisce i livelli 3 e 4, così come il numero di porta unisce i livelli 4 e 5

- *checksum dell'header*: aiuta il router a individuare bit errati in un datagram ricevuto. Il router calcola il checksum dell'header per ogni datagram ricevuto e lo confronta con quello contenuto nel campo: se li trova diversi, definisce la condizione di errore e scarta il datagram
- *source IP address* e *destination IP address*: quando il mittente crea un datagram, inserisce il proprio indirizzo IP nel campo *source IP address* e l'indirizzo del destinatario nel campo *destination IP address*
- *opzioni*: questo campo, che consente di estendere la lunghezza dell'header, è causa di complicazioni insite nell'IPv4. La lunghezza variabile dell'header, infatti, non permette di determinare a priori dove iniziano i dati trasportati. Poiché alcune opzioni possono richiedere elaborazioni da parte dei router, il tempo di transito del datagram può variare considerevolmente
- *dati*: il campo contenente i dati da trasportare, generalmente il segmento TCP o UDP del livello trasporto. Questo campo, tuttavia, può contenere altri tipi di dati, ad esempio i messaggi ICMP

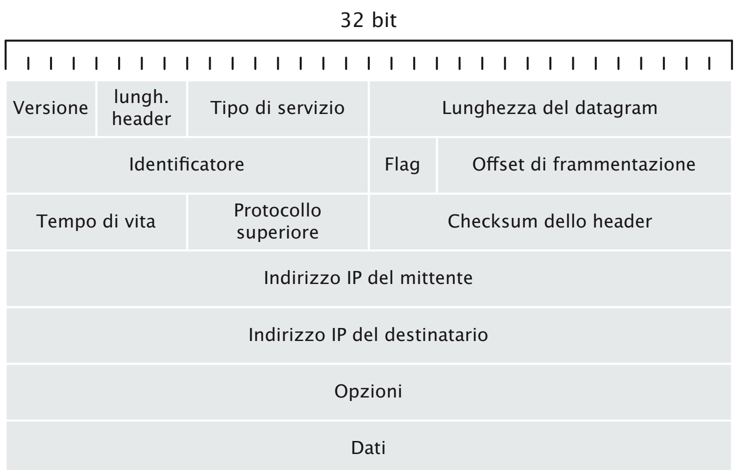


Figura **10**
Formato di un datagram IPv4.

Frammentazione • Vedremo nel prossimo capitolo come non tutti i protocolli del livello connessione siano in grado di trasportare pacchetti del livello rete delle stesse dimensioni: l'Ethernet, ad esempio, può portare frame di 1.500 byte, mentre alcune connessioni WAN non superano i 576 byte.

La massima quantità di dati che un frame può trasportare si dice *massima unità di trasmissione* o **MTU** (*Maximum Transmission Unit*). Poiché i datagram IP sono incapsulati nei frame per essere trasportati da un router all'altro, l'MTU pone un grosso limite alle loro dimensioni, anche considerando il fatto che ogni tratta può utilizzare un protocollo di livello connessione diverso, e quindi con diversa MTU.

Supponiamo che un router riceva un datagram da una connessione con MTU elevata, e debba instradarlo su una connessione con MTU inferiore: l'unica soluzione possibile è frammentare i dati su due o più datagram di dimensioni minori, e inviarli incapsulati in più frame separati: ognuno di questi datagram più piccoli è chiamato **frammento**.

Per evitare che i router si debbano sobbarcare un compito piuttosto pesante e per mantenere quanto più possibile semplice la struttura della rete, il riassemblaggio del segmento a partire dai frammenti contenuti nei datagram viene effettuato dall'host destinatario. Per fare ciò, l'host deve sapere se il datagram che riceve è un frammento, quanti sono i frammenti in cui è stato suddiviso il datagram originale e qual è l'ordine in cui i frammenti vanno riassemblati: queste informazioni sono contenute nei campi *identificatore*, *flag* e *offset di frammentazione*.

Il primo è un identificativo univoco che il mittente inserisce insieme agli indirizzi IP,

tipicamente incrementandolo di 1 a ogni datagram successivo: in questo modo, quando un router frammenta il datagram, i datagram risultanti mantengono tutti lo stesso identificativo. Quando il destinatario riceve la sequenza di datagram, esaminando l'identificatore stabilisce se un datagram è un frammento di un pacchetto più grande.

Poiché IP fornisce un servizio non affidabile, e quindi alcuni pacchetti possono andare persi, il destinatario deve sapere quando riceve l'ultimo frammento della sequenza. Per questa ragione, il router marca il campo flag dell'ultimo frammento con uno 0, mentre tutti gli altri frammenti hanno il campo a 1.

Sempre a causa dell'inaffidabilità di IP, è necessario che il destinatario sia in grado di vedere in che ordine sono stati generati i frammenti ricevuti e se qualcuno si è perso nella trasmissione: il campo offset, riempito dal router, provvede a fornire queste informazioni, indicando la somma dei byte contenuti in tutti i frammenti precedenti.

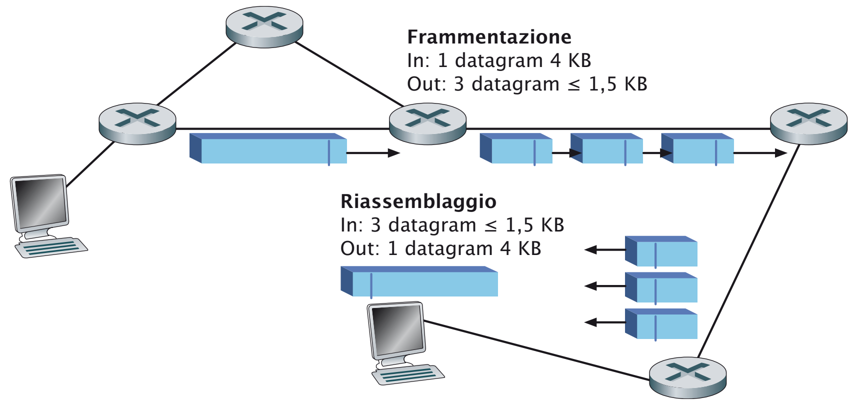


Figura **11** Frammentazione e riassemblaggio di un datagram.

La frammentazione IP gioca un ruolo fondamentale nella capacità dell'Internet di far convivere tecnologie disparate a livello connessione, ma ha anche un costo in termini di complicazione e di vulnerabilità, dato che richiede funzioni complesse di suddivisione nei router e di ricostruzione negli host; inoltre, la semplicità del sistema permette attacchi letali sia alla rete, sia al sistema operativo degli host.

Indirizzamento IPv4 • Consideriamo come gli host e i router sono connessi all'interno di una rete. Un host ha, tipicamente, una sola connessione alla rete e quando l'IP nell'host vuole inviare un datagram, lo fa attraverso di essa: il confine tra l'host e la connessione fisica della rete si chiama interfaccia.

Un router ha il compito di instradare i datagram in direzioni diverse, quindi possiede più connessioni e, di conseguenza ha più interfacce, una per ogni connessione. Poiché ogni interfaccia è associata a una connessione diversa, IP richiede che ognuna di esse abbia un proprio indirizzo: l'indirizzo IP, quindi, è tecnicamente associato a un'interfaccia, non all'apparato che la contiene.

Ogni indirizzo IP è formato da 4 byte, ossia 32 bit, e quindi sono disponibili 4.294.967.296 indirizzi diversi. Per semplicità, gli indirizzi sono normalmente scritti in notazione decimale puntata (ossia ogni byte è indicato nella sua forma decimale, separato dagli altri byte con un punto). Gli indirizzi, quindi, vanno da 0.0.0.0. a 255.255.255.255. Vedremo più avanti che questi due indirizzi estremi hanno un utilizzo particolare e non possono essere assegnati a un'interfaccia.

Gli indirizzi IP, però, non possono essere accoppiati alle interfacce in maniera casuale, altrimenti ogni tabella di routing dovrebbe comprendere tutti gli indirizzi. Per evitare questo problema, che porterebbe l'instradamento a una lentezza inaccettabile, gli indirizzi vengono accorpati in gruppi che presentano parte dei byte, o dei bit all'interno del byte, identici e vengono assegnati a interfacce di host e router direttamente connessi tra loro, che vengono così a formare una sottorete. Facciamo un esempio.

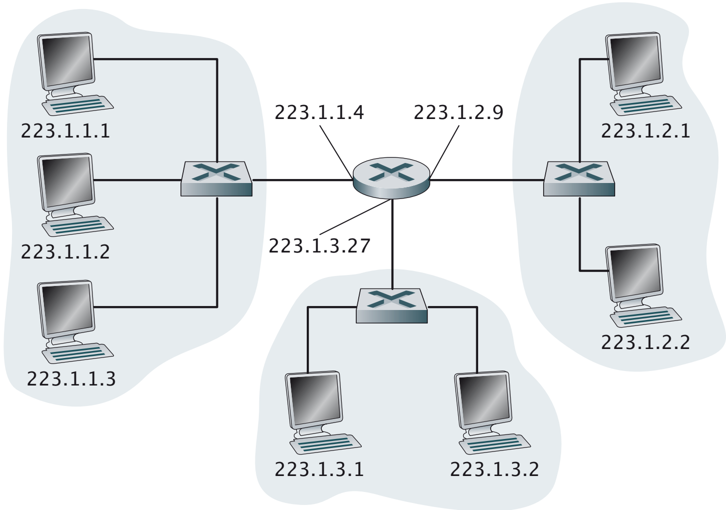


Figura **12**
Indirizzamento IPv4.

Nella figura 12 vediamo un certo numero di host connessi tra loro e a tre diverse porte di un router attraverso un apparato che non gestisce il livello rete, tipicamente uno switch, e che quindi non è dotato di indirizzo IP. Tutte le interfacce hanno i primi due byte identici, ma ogni porta del router e gli host a essa collegati hanno il terzo byte diverso: le interfacce che hanno il terzo byte identico formano una *sottorete*. All'interno di ogni sottorete, le diverse interfacce sono distinte dal quarto byte, che è assegnato univocamente.

Maschera di sottorete • In questo modo, l'indirizzamento IP può assegnare un indirizzo univoco all'intera sottorete, che viene espresso aggiungendo ai quattro byte un ulteriore numero, detto *maschera di sottorete*, che indica il numero di bit, partendo da sinistra, che fanno parte dell'indirizzo.

Nel caso in figura 12, abbiamo tre sottoreti: 223.1.1.0/24, 223.1.2.0/24, 223.1.3.0/24, dove 24 indica il numero di bit da prendere in considerazione. All'interno della sottorete 223.1.1.0/24, quindi, è possibile assegnare a ogni interfaccia tutti gli indirizzi con i primi tre byte 223.1.1. e gli ultimi otto bit compresi tra 0 e 255.

Per comodità, la maschera di sottorete viene normalmente indicata, nei pannelli di configurazione, nella forma decimale: nel caso delle sottoreti in figura 12, i 24 bit sono scritti 255.255.255.0. Se i bit fossero stati venti, la maschera di sottorete sarebbe diventata 255.255.240.0; se fossero stati sedici, sarebbe diventata 255.255.0.0.

Dalla definizione appena data, vediamo che anche le connessioni tra due router devono essere considerate sottoreti: quindi, anche all'interno dell'Internet viene applicata la stessa strategia di indirizzamento, chiamata **CIDR** (*Classless Interdomain Routing*), che generalizza la nozione di indirizzamento di sottorete.

In questo modo è possibile suddividere la rete in macroblocchi, ognuno dei quali è a sua volta suddiviso in blocchi più piccoli, e così via sino a giungere alle sottoreti più piccole. Gli x bit più significativi della forma a.b.c.d/x costituiscono la porzione di rete dell'indirizzo IP e vengono normalmente denominati *prefisso dell'indirizzo* stesso.

Un esempio • Poniamo che un ISP disponga di un blocco di indirizzi 233.1.0.0/16 e che debba suddividerli tra più organizzazioni ad esso collegate, ciascuna delle quali, a sua volta, presenta delle sottoreti interne.

L'ISP può assegnare a ogni organizzazione un blocco di indirizzi con prefisso più lungo, ad esempio 233.1.240.0/20.

Questo significa che l'organizzazione può assegnare al proprio interno tutti gli indirizzi che hanno il terzo byte compreso tra 240 e 255, quindi generare 16 sottoreti, con indirizzi 233.1.240.0/24, 233.1.241.0/24 e via così fino a 233.1.255.0/24.

Ma ognuna di queste sottoreti può a propria volta essere suddivisa, aumentando il numero

di bit del prefisso: ad esempio, è possibile generare la sottorete 233.1.240.0/28, che consente l'indirizzamento di 16 interfacce, seguita dalla sottorete 233.1.240.16/28 per altre 16 interfacce e dalla 233.1.240.128/25 per 128 interfacce.

Gli indirizzi IP sono gestiti dall'**ICANN** (*Internet Corporation for Assigned Names and Numbers*), che ha anche il compito di gestire i root server DNS, di cui abbiamo parlato nel capitolo 1. A propria volta, l'ICANN assegna gli indirizzi ai registri Internet regionali, che provvedono all'assegnazione e gestione degli indirizzi nelle aree di propria competenza.

DHCP • Una volta che un'organizzazione ha ottenuto un blocco di indirizzi, deve assegnarli a ogni host e router al proprio interno. Ovviamente, l'assegnazione può essere fatta manualmente dall'amministratore di rete, intervenendo sui pannelli di configurazione degli apparati ma, a meno di particolari necessità, questo compito viene solitamente svolto dal protocollo **DHCP** (*Dynamic Host Configuration Protocol*), che permette a un host di ottenere un indirizzo IP automaticamente quando si connette alla rete.

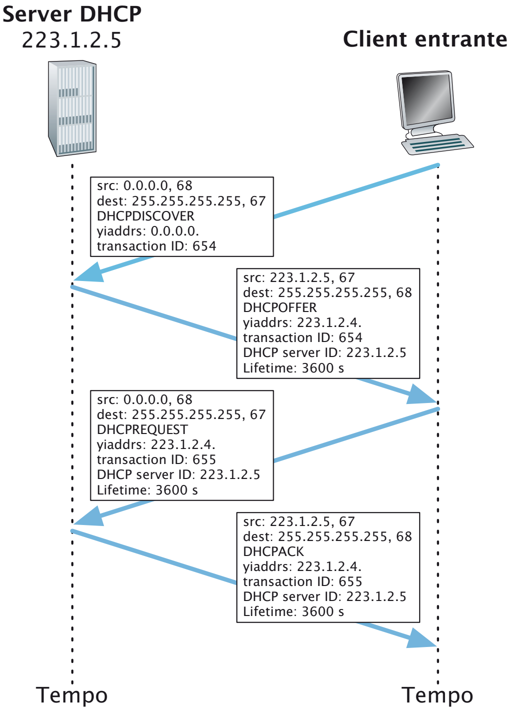


Figura **13**
Assegnazione di un indirizzo IP tramite DHCP.

Questo protocollo può essere configurato in modo che un host riceva sempre lo stesso indirizzo IP, oppure che questo possa cambiare ogni volta che l'host si connette alla rete. Oltre all'assegnazione degli indirizzi, DHCP provvede a fornire all'host la maschera di sottorete, l'indirizzo del router a cui è connesso e quello del server DNS locale.

DHCP viene usato correntemente nelle reti domestiche di accesso all'Internet e nelle WLAN, in cui gli host entrano ed escono frequentemente dalla rete: in questo modo, dato un numero di utenti *N*, che però non si collegano mai contemporaneamente in più di *M* (inferiore a *N*), il numero di indirizzi da assegnare sarà *M* e non *N*, consentendo così una maggiore efficienza del sistema, che non si trova allocati *N-M* indirizzi inutilizzati.

DHCP è un protocollo client-server. Il client è tipicamente un host che cerca di accedere alla rete e vuole ottenere le informazioni di configurazione, compreso un proprio indirizzo IP. Nel caso più semplice, ogni sottorete è dotata di un proprio server DHCP, ma spesso ciò non avviene e si utilizza un **relay agent** (*agente ripetitore*), tipicamente un router, che conosce l'indirizzo del server, esterno alla sottorete, che ne gestisce il DHCP. Il processo di allocazione avviene in quattro passi (figura 13):

- *DHCP discovery*: quando l'host si collega alla rete, cerca un server DHCP con cui interagire, inviando sulla porta 67 un messaggio di richiesta all'interno di un pacchetto UDP, che a sua volta è incapsulato in un datagram. L'host non conosce l'indirizzo del server - altrimenti non lo cercherebbe - e non ha a sua volta un indirizzo proprio a cui ricevere la risposta: entrano in gioco allora i due indirizzi IP che abbiamo visto prima essere riservati: 255.255.255.255. e 0.0.0.0. Il primo è detto *indirizzo broadcast* e, inserito nel campo *destination IP*, causa la propagazione del messaggio a tutti gli host della rete; il secondo viene inserito dall'host nel campo *source IP* come proprio indirizzo provvisorio
- *DHCP offer*: tutti i server DHCP presenti nella rete riservano all'host un indirizzo e lo comunicano a tutti i nodi della sottorete, sempre con l'indirizzo broadcast. Il messaggio contiene l'identificativo dell'host chiamante, l'indirizzo proposto, la maschera di sottorete e il tempo di concessione dell'indirizzo, ossia la durata di validità dell'indirizzo. È possibile che l'host riceva una proposta da più di un server DHCP, nel qual caso è sua facoltà scegliere quale di esse accettare
- *DHCP request*: una volta scelta l'offerta, l'host risponde al server, di cui adesso conosce l'indirizzo IP, con un messaggio di richiesta, ripetendo i parametri di configurazione

- *DHCP ack*: il server risponde con un messaggio di accettazione, confermando i parametri richiesti

Una volta che il client ha ricevuto il messaggio di accettazione, la sequenza è completa e l'host è connesso in rete. Nel caso il client voglia utilizzare l'indirizzo oltre i termini della concessione, il protocollo DHCP prevede un meccanismo di rinnovo da parte del client.

È chiaro che il protocollo DHCP è l'unica possibilità da parte di un host di passare da una sottorete a un'altra senza bisogno di riconfigurare manualmente la propria connessione. Inoltre permette l'assegnazione automatica di indirizzi in ambienti domestici o in piccole sottoreti, dove, sicuramente, non è presente la figura dell'amministratore di sistema.

Tuttavia, DHCP presenta dei difetti in caso di connessione da dispositivi mobili: a ogni cambio di cella, infatti, è necessario rinegoziare l'indirizzo IP, dato che cambia la sottorete, e ciò comporta la caduta della connessione. Per eliminare il problema, è stata realizzata un'estensione al protocollo, in modo da permettere a un nodo mobile di mantenere lo stesso indirizzo anche muovendosi tra le sottoreti.

NAT • Abbiamo visto che qualunque interfaccia richiede un indirizzo IP, quindi tutte le porte di tutti i router che compongono l'Internet e tutti gli host che vi accedono ne devono essere dotate.

Benché il numero di indirizzi disponibili sia molto alto, non è infinito e, con la grande crescita di reti domestiche e di dispositivi mobili connessi, è necessario eliminare qualunque spreco.

Quando viene creata una LAN domestica o di ufficio, l'ISP cui è connessa deve fornire un numero di indirizzi corrispondente al numero di macchine che compongono la sottorete; se, però, la sottorete cresce, non è detto che l'ISP abbia a disposizione degli indirizzi contigui da allocare, dato che possono essere stati assegnati a un'altra sottorete. Questo comporta l'impossibilità di ampliare la sottorete, oppure la necessità di mantenere liberi un certo numero di indirizzi IP contigui per future possibili esigenze: nel primo caso si fornisce un cattivo servizio, nel secondo si sprecano risorse.

La soluzione, ormai correntemente utilizzata in queste situazioni, si chiama **NAT** (*Network Address Translation*).

Il NAT consiste nella traduzione automatica, da parte di un router abilitato, dell'indirizzo IP dei pacchetti che transitano attraverso le sue porte: in questo modo gli indirizzi IP della sottorete - detti indirizzi IP privati - non vengono visti dall'esterno, quindi non entrano nel numero degli indirizzi allocati.

Grazie a questa tecnica, è possibile assegnare a tutte le sottoreti lo stesso blocco di indirizzi IP, eliminando gli sprechi visti prima. Queste sottoreti, che hanno indirizzi IP visibili solo al loro interno, vengono dette *reti private*. IANA ha riservato alle reti private tre gruppi di indirizzi IP:

- **10.0.0.0/8** (gli indirizzi compresi tra 10.0.0.0. e 10.255.255.255)
- **172.16.0.0/12** (gli indirizzi compresi tra 172.16.0.0. e 172.31.255.255)
- **172.178.0.0/16** (gli indirizzi compresi tra 172.178.0.0. e 192.168.255.255)

Attualmente, esistono centinaia di migliaia di reti private che utilizzano il sottoinsieme di indirizzi 10.0.0.0/24. e quindi l'Internet non può utilizzare questi indirizzi.

Per questa ragione, il router che realizza il NAT viene visto dall'esterno come un host con un singolo indirizzo IP - detto indirizzo IP pubblico - e tutte le comunicazioni con l'intera sottorete passano per quest'unico indirizzo: in questo modo, però, non è possibile distinguere quale degli host nella sottorete sia il vero destinatario di una connessione, visto che l'intera sottorete possiede apparentemente un unico indirizzo IP.

La soluzione del problema consiste nella tecnica detta di **port forwarding**, che permette a un utente esterno di raggiungere un host con indirizzo IP privato mediante una porta predefinita dell'IP pubblico dello stesso, ossia del router che effettua il NAT.

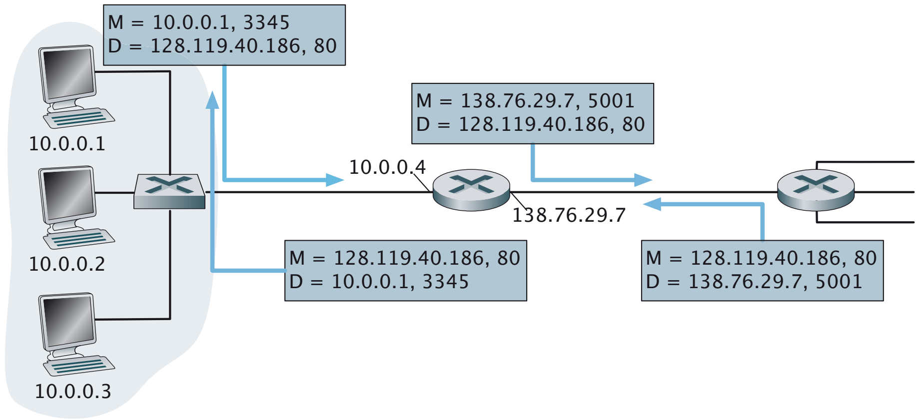


Figura **14**
Funzionamento del NAT.

Facendo riferimento alla figura 14, vediamo che tutto il traffico verso l'Internet ha il source address 138.76.29.7, e tutto il traffico entrante nella sottorete deve avere lo stesso destination address.

Per ripartire il traffico tra i vari host, all'interno del router esiste una tabella di conversione che, oltre agli indirizzi IP, associa anche una specifica porta dell'host a una del router, diversa da quelle riservate a applicazioni a funzioni di livello più elevato: nel nostro caso, la porta 3345 dell'host 10.0.0.1. è associata alla porta 5001 del router.

Poniamo di voler inviare una richiesta al Web server 128.119.40.186, che sarà indirizzata alla sua porta 80.

Il pacchetto raggiungerà il router con source address 10.0.0.1, 3345 e destination address 128.119.40.186, 80.

Il router verifica nella tabella di conversione che alla porta 3345 dell'host corrisponde la propria porta 5001 e invia il pacchetto al Web server con source address 138.76.29.7, 5001 e destination address invariato.

Il Web server risponde con un pacchetto che ha destination address 138.76.29.7, 5001, ossia la connessione WAN del router, il quale provvede, sempre sulla base della tabella di conversione, a inoltrarlo all'host 10.0.0.1, 3345.

Il NAT viola alcune regole fondamentali del modello ISO-OSI: in primo luogo, le porte dovrebbero indirizzare processi e non host; inoltre, un router non dovrebbe processare pacchetti oltre il livello 3; in terzo luogo, viola il cosiddetto argomento end-to-end, ossia la regola per cui gli host dovrebbero parlare direttamente tra loro, senza che i nodi intermedi modifichino in alcun modo indirizzi e porte.

Tuttavia, questa tecnica è l'unica che permette di moltiplicare le sottoreti senza occupare troppi indirizzi IP. Il problema si risolverà con l'utilizzo generalizzato dell'IPv6, ma il percorso da fare è ancora lungo: vedremo più avanti perché.

ICMP • Come dice il suo nome, ICMP (*Internet Control Message Protocol*), è un protocollo usato da host e router per comunicare tra loro informazioni riguardanti il livello rete. Dal punto di vista dell'architettura, ICMP giace immediatamente sopra IP, dato che è trasportato all'interno dei datagram, esattamente come TCP o UDP.

I messaggi ICMP comprendono un campo *tipo* e un campo *codice*, e contengono l'header e i primi otto byte del datagram che ha causato la generazione del messaggio: notiamo, guardando la tabella dei messaggi più comuni, che ICMP non porta solo messaggi di errore.

Tabella **1**
I messaggi ICMP e i campi che costituiscono il protocollo.

Tipo ICMP	Codice	Descrizione
0	0	echo reply
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unreachable
4	0	source quench (controllo di congestione)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

È interessante il messaggio 4-0, source quench (*controllo di congestione*), il cui scopo iniziale era consentire a un router in congestione di inviare un messaggio agli host per farli rallentare e, quindi, permettergli di smaltire le code.

Poiché, però, TCP ha un proprio meccanismo di controllo della congestione che opera a livello più alto, il source quench è molto poco utilizzato.

IPv6 • All'inizio degli anni '90, lo IETF ha iniziato a sviluppare il successore del protocollo IPv4, prevedendo che l'indirizzamento a 32 bit si sarebbe, nel tempo, rivelato insufficiente. In realtà la crescita è stata più veloce del previsto, e i tempi di attuazione della transizione si sono ulteriormente accorciati.

La decisione di generare una nuova versione del protocollo IP aveva anche lo scopo di eliminare alcune criticità emerse dell'esperienza di utilizzo.

Una versione intermedia IPv5, che avrebbe dovuto essere l'ufficializzazione di un protocollo chiamato **ST II** (*Stream Protocol II*), è stata però cassata ed è entrata a far parte della struttura di rete ATM e MPLS.

Il datagram IPv6 • Il formato del datagram IPv6 è molto diverso dal formato IPv4, ed è incompatibile con esso. Le principali differenze sono:

- *maggiori capacità di indirizzamento*: anziché 32 bit ne sono disponibili 128. Con questa modifica si rende disponibile una quantità di indirizzi IP (precisamente 2^{128}, pari a un numero a 38 cifre), che sarà sufficiente per qualunque sviluppo futuro
- *un header snellito*: sono stati eliminati alcuni campi dell'IPv4 e se ne è fissata la lunghezza a 40 byte, in modo da poterlo processare più velocemente
- *etichettatura dei flussi e priorità*: permetterà, in futuro, di gestire i datagram provenienti da host diversi in modo personalizzato

Il datagram IPv6 è composto dai seguenti campi (figura 15):

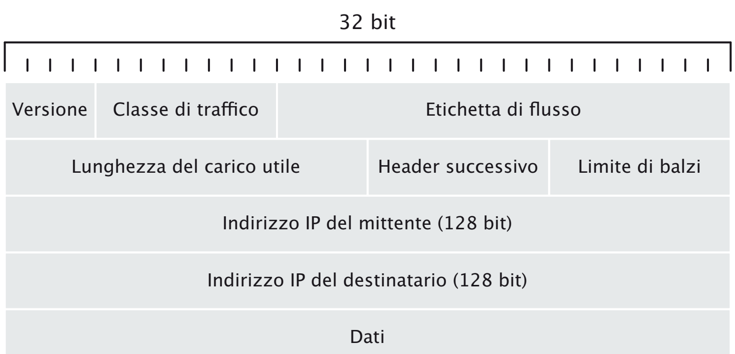


Figura **15**
Formato di un datagram IPv6.

- *numero di versione*: dichiara la versione del protocollo IP che si sta usando: leggendolo, il router determina come interpretare il resto del datagram
- *classe di traffico*: sostituisce il campo TOS dell'IPv4 e ne condivide le funzioni
- *etichetta di flusso*: ha lo scopo di identificare un flusso di datagram da un altro
- *lunghezza del carico utile*: riporta il numero di byte contenuti dal datagram a valle dell'header
- *header successivo*: identifica il protocollo di livello superiore cui i dati contenuti nel datagram saranno consegnati, ad esempio TCP. Corrisponde al campo *protocollo* di IPv4
- *limite di balzi*: il contenuto di questo campo subisce il decremento di un'unità a ogni passaggio attraverso un router; quando giunge a 0, il datagram viene eliminato. Sostituisce il campo *tempo di vita* dell'IPv4
- *source IP address* e *destination IP address*: come già detto, anziché avere una lunghezza di 32 bit, hanno lunghezza 128 bit
- *dati*: ha lo stesso contenuto del campo corrispondente del protocollo IPv4

Notiamo che non sono più presenti nell'IPv6 alcuni campi dell'header IPv4, e precisamente:

- *identificatore, flag, offset di frammentazione*: nel protocollo IPv6, le operazioni di frammentazione e riassemblaggio avvengono solo presso il mittente ed il destinatario, per cui non sono più svolte dai router intermedi. Se il datagram ricevuto da un router non può essere instradato sulla connessione successiva, il router lo scarta e invia all'host un messaggio di errore ICMP di "*pacchetto troppo grande*". In questo modo, il mittente può reinviare i dati in datagram più piccoli. Poiché frammentazione e riassemblaggio richiedono tempi significativi, la loro collocazione solo sugli host rende molto più veloce l'instradamento lungo la rete.
- *checksum dell'header*: anche in questo caso, lo scopo è di diminuire i tempi di transito. Nell'IPv4 l'header contiene un campo TTL che viene decrementato a ogni passaggio, e quindi il checksum deve essere ricalcolato da ogni router. D'altra parte, poiché i protocolli di livello superiore hanno già funzioni di checksum, si è ritenuto che nell'IPv6 questo fosse ridondante
- *opzioni*: questo campo è stato eliminato per mantenere una lunghezza fissa dell'header. Le opzioni vengono trattate come header successivi, alla stessa stregua degli header di livello superiore, e quindi sono contenute nel campo dati

La forma in cui l'indirizzo IPv6 è rappresentato varia considerevolmente rispetto all'IPv4: l'uso della notazione decimale puntata, infatti, porterebbe a una sequenza di lunghezza ingestibile.

Si è perciò deciso do utilizzare una notazione esadecimale dividendo l'indirizzo in 8 gruppi di 16 bit, divisi dai due punti. Tenendo conto che già i primi 64 bit consentono un indirizzamento di oltre 16 miliardi di miliardi, la seconda metà dell'indirizzo, sicuramente tutta a zero, può essere tralasciata. Un indirizzo IPv6, quindi, può essere scritto così:

2001:0DB8:AC10:FE01

che corrisponde a

0010000000000001:0000110110111000:1010110000010000:1111111000000001

4 Algoritmi e protocolli di routing

Abbiamo visto che gli algoritmi di routing, operando all'interno dei router, scambiano ed elaborano le informazioni che servono per configurare le tabelle di inoltro. Sia che si tratti di un servizio datagram o di un servizio a circuito virtuale, in ogni caso il livello rete deve determinare il percorso che i pacchetti devono fare tra il mittente e il destinatario, ossia definire i percorsi tra i router della rete.

Tipicamente, un host è connesso direttamente a un solo router, chiamato *router di default*: quando un host invia un pacchetto, questo è trasferito sempre al router di default. Se chiamiamo *source router* il router di default del mittente e *destination router* il router di default del destinatario, il problema di trasferire un pacchetto dal mittente al destinatario diventa, quindi, quello di trasferire il pacchetto dal source router al destination router.

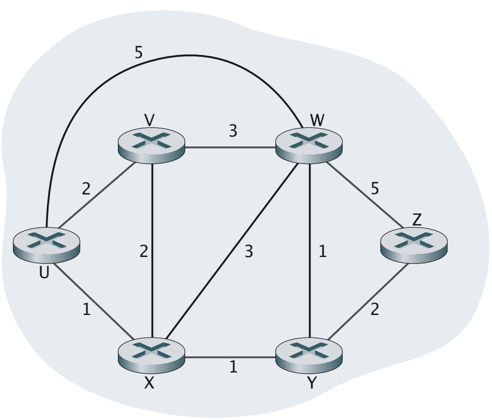


Figura **16**
Rappresentazione come grafo di una rete. La rappresentazione di una rete mediante un grafo, in cui i nodi corrispondono a router e gli archi a tratte fisiche, mette in evidenza efficacemente la struttura fisica della rete stessa.

Lo scopo di un algoritmo di routing è, quindi, apparentemente semplice: dato un gruppo di router interconnessi in modo ridondante, l'algoritmo deve trovare il percorso ottimale tra source router e destination router.

Tipicamente, il percorso ottimale è quello a minor costo, ma nella pratica alcune questioni di natura politica (ad esempio il divieto per i router di una certa organizzazione X di instradare pacchetti provenienti dai router di un'altra organizzazione Y) complicano notevolmente il lavoro degli algoritmi.

Percorso ottimale • Il grafo di figura 16 mostra una rete con 6 nodi e 10 archi. Per ogni arco, è indicato anche il costo di transito, in modo da poter calcolare quale sia il percorso ottimale per ogni connessione tra due nodi qualunque.

Ad esempio, volendo calcolare il percorso tra U e W, abbiamo tre possibilità, coinvolgendo il minor numero di nodi: UW diretto, UV + VW e UX + XW. Il primo percorso ha costo 5, il secondo 5 e il terzo 4: quindi il percorso ottimale sembra essere UX+XW. In realtà, se esaminiamo anche il costo di percorsi con un maggior numero di nodi, troviamo che il percorso ottimale è UX + XY + YW, il cui costo totale è 3.

Da questo semplice esempio si capisce come sia complesso generare algoritmi di routing efficienti in reti che comprendono centinaia o migliaia di router.

Il costo di una tratta, inoltre, può dipendere da molti fattori: la lunghezza della tratta stessa, il mezzo trasmissivo, la velocità della connessione, il suo grado di occupazione. In ogni caso, anche la determinazione del costo da far utilizzare dagli algoritmi è complessa e può presentare molte incognite.

Tipi di algoritmi • Possiamo dividere grossolanamente gli algoritmi di routing in due categorie, *globali* e *decentrati*:

- un *algoritmo globale* calcola il percorso ottimale tra mittente e destinatario utilizzando una conoscenza globale della rete, ossia prende in considerazione tutte le connessioni possibile e ne conosce i costi. Ciò significa che l'algoritmo deve ottenere tutte le informazioni prima di procedere al calcolo. Volendo calcolare in questo modo la connessione tra U e Z nella figura 16, che ha solo sei nodi, dovremmo già prendere in considerazione 17 percorsi diversi
- in un *algoritmo decentrato* il calcolo del percorso ottimale è effettuato in modo *distribuito* e *iterativo*. Nessun nodo possiede le informazioni sui costi di tutte le tratte, ma solo di quelli relativi alle proprie connessioni dirette. Attraverso un processo iterativo di calcolo e scambio di informazioni con i nodi vicini, l'algoritmo calcola il percorso ottimale. Se pensiamo a come abbiamo calcolato i percorsi nella rete di figura 16, vediamo che questo metodo è quello che usiamo mentalmente

Un'ulteriore classificazione degli algoritmi di routing li distingue in *statici* e *dinamici*. Un algoritmo è statico quando i percorsi cambiamo molto lentamente nel tempo, spesso per l'intervento di modifiche alla struttura di rete; un algoritmo è dinamico quando cambia il routing in risposta ai carichi di traffico e ai cambiamenti topologici.

La trattazione delle caratteristiche degli algoritmi di routing è dei relativi protocolli utilizzati nell'Internet è argomento complesso e di non semplice apprendimento. Un approfondimento in merito è disponibile online.

A4-02
Algoritmi di routing

Concetti essenziali

- Le funzioni principali del livello rete sono l'inoltro e l'instradamento.
- L'inoltro consiste nel trasferimento, all'interno di un router, di un pacchetto da un dato ingresso a una determinata uscita.
- L'instradamento è la definizione, ottenuta attraverso una particolare strategia, del percorso complessivo del pacchetto dal mittente al destinatario.
- La rete di accesso connette fisicamente un host al primo router dell'Internet.
- Un circuito virtuale consiste in un percorso tra mittente e destinatario, in cui sono specificati numeri e valori nelle tabelle di inoltro dei router.
- Un circuito virtuale viene instaurato, utilizzato per il trasferimento dei dati e annullato al compimento della trasmissione.
- In una rete datagram, se un host vuole spedire un pacchetto, lo marca con l'indirizzo di destinazione e lo spinge (*push*) sulla rete stessa.
- L'architettura di un router può essere vista come composta da porte di ingresso, porte di uscita, processore di routing e switching fabric.
- In un router, lo switching fabric è il vero elemento di commutazione: esso può essere realizzato attraverso commutazioni via memoria, via bus o via matrice.
- Gli elementi nei quali si possono creare addensamenti di traffico derivanti dall'incapacità di smaltimento corretto nel tempo dovuto dei pacchetti prendono il nome di colli di bottiglia.
- Il protocollo IP esiste in due versioni: IPv4 e IPv6.
- Le due versioni del protocollo IP differiscono per l'indirizzamento, molto più esteso in IPv6, e per la struttura dei rispettivi datagram.
- Attraverso un DHCP (*Dynamic Host Configuration Protocol*) un host ottiene un indirizzo IP automaticamente quando si collega alla rete.
- I percorsi di routing vengono calcolati attraverso particolari algoritmi che determinano i tragitti ottimali, in funzione dei parametri di merito che si intendono applicare.
- Esistono algoritmi di costruzione del percorso ottimale di tipo globale e di tipo decentrato.

Test

1 Dire se le seguenti affermazioni sono vere o false.
Il livello rete provvede a:
- A inserire i blocchi provenienti dal livello 1 ed inviarli al livello 5 V F
- B inviare i datagram al livello superiore V F
- C prendere i blocchi del livello superiore e inserirli in datagram V F
- D verificare la congruenza dei datagram eliminando quelli errati V F

2 Dire se le seguenti affermazioni sono vere o false.
A livello rete, lo spostamento dei pacchetti dall'host mittente a quello destinatario avviene attraverso:
- A inoltro e instradamento V F
- B ricezione e traduzione V F
- C store and forward V F
- D cattura e rilascio V F

3 Dire se le seguenti affermazioni sono vere o false.
Nel router troviamo:
- A tavole di interscambio V F
- B tabelle di inoltro V F
- C tabelle di associazione V F
- D tavole di congiunzione V F

4 Dire se le seguenti affermazioni sono vere o false.
In un router, si dice switching fabric l'elemento che:
- A connette le porte di ingresso del router a quelle di uscita V F
- B protegge l'alimentatore dalle sovratensioni V F
- C trasforma le porte di ingresso in porte di uscita V F
- D verifica il contenuto dei pacchetti in transito V F

5 Dire se le seguenti affermazioni sono vere o false.
I seguenti sono tutti modelli di servizio forniti dal livello rete:
- A consegna garantita con ritardo limite definito, autocorrezione, zero delay V F
- B banda massima garantita, banda autoconfigurante, consegna garantita V F
- C consegna differita, anticipo all'ordine, rimodulazione

del ritardo V F

D banda minima garantita, consegna ordinata, consegna garantita V F

6 **Dire se le seguenti affermazioni sono vere o false.**

In una rete datagram, se l'host vuole inviare un pacchetto:

A lo marca con l'indirizzo del destinatario e attende l'ok dal centro V F

B non può farlo se non ha da trasmettere un numero minimo di pacchetti V F

C deve richiedere l'autorizzazione all'host destinatario V F

D lo spinge nella rete dopo averlo marcato con l'indirizzo del destinatario V F

7 **Dire se le seguenti affermazioni sono vere o false.**

In una rete datagram, le tabelle di inoltro:

A sono configurate in fabbrica con valori predefiniti V F

B si mantengono inalterate dall'1/1 al 31/12 e vengono mutate ogni 1 gennaio per ragioni di sicurezza V F

C possono essere riconfigurate dagli utenti finali V F

D vengono modificate dagli algoritmi di routing in continuazione V F

8 **Dire se le seguenti affermazioni sono vere o false.**

Nello switching fabric la commutazione può avvenire:

A via memoria V F

B via bus V F

C via check-in V F

D via router V F

9 **Dire se le seguenti affermazioni sono vere o false.**

In un processo di comunicazione, il punto che risulta più critico in termini di velocità di inoltro dei dati si dice:

A critical node V F

B manico di caraffa V F

C collo di bottiglia V F

D bottleneck V F

10 **Dire se le seguenti affermazioni sono vere o false.**

I tre componenti principali del livello di rete, dal punto di vista dei protocolli, sono:

A IP, fasting, protocolli di delivery V F

B ICMP, IP, protocolli di routing V F

C ICMP, ADSL, DSL V F

D protocolli di apertura, protocolli di chiusura, GIMP V F

11 **Indicare quali delle seguenti affermazioni (una o più di una) sono vere.**

I datagram IP per essere trasportati da un router all'altro devono essere:

A incapsulati in un frame V F

B inviati periodicamente V F

C modificati in funzione della velocità di rete V F

D convertiti in file .txt V F

12 **Dire se le seguenti affermazioni sono vere o false.**

I seguenti sono indirizzi di rete validi:

A 255.12.34.128 V F

B 232.111.456.2 V F

C 208.208.208.208 V F

D 129.0.0.0 V F

13 **Dire se le seguenti affermazioni sono vere o false.**

DHCP significa:

A Dynamic High Control Protocol V F

B Dual Host Check Point V F

C Dynamic Host Configuration Protocol V F

D Destination Halt Check Position V F

14 **Dire se le seguenti affermazioni sono vere o false.**

Grazie al DHCP, un host che si connette in rete:

A ottiene automaticamente un indirizzo IP V F

B ottiene automaticamente la massima banda V F

C dichiara il proprio indirizzo IP alla rete V F

D dichiara di voler operare in rete senza indirizzo IP V F

15 **Dire se le seguenti affermazioni sono vere o false.**

Grazie al NAT:

A i pacchetti che arrivano dall'esterno vengono bloccati all'ingresso di una sottorete V F

B tutti i pacchetti della sottorete vengono immessi automaticamente nella rete esterna V F

C i pacchetti destinati alla sottorete non vengono passati all'esterno V F

D ogni sottorete può scegliere quali pacchetti accettare e quali no V F

16 **Dire se le seguenti affermazioni sono vere o false.**

In un algoritmo di routing decentrato:

A nessun nodo conosce tutti i costi di tutte le tratte V F

B nessun nodo conosce i costi di alcuna tratta V F

C ogni nodo conosce i costi solo delle tratte ad esso vicine V F

D ogni nodo può negoziare i costi di tratta sulla base dei valori di mercato V F

5 Il livello connessione

Nella comunicazione di rete, il livello connessione svolge una serie di attività fondamentali per l'inoltro dei pacchetti tra la sorgente e la destinazione. Il capitolo prende in esame i principali compiti del livello connessione: dall'incapsulamento di ogni datagram in un frame, all'accesso alla connessione, al rilevamento e al tentativo di correzione automatica degli errori che si fossero verificati nel corso dell'invio.

Pur affrontando in dettaglio lo specifico tema in queste pagine, ulteriori approfondimenti tecnici sono stati resi disponibili sul sito Web a corredo del volume, dando facoltà al docente e agli allievi di esplorare, se desiderato, in maggior profondità l'argomento.

1 Introduzione

Dopo aver visto come il livello rete fornisce un servizio di comunicazione tra due host, passando attraverso un certo numero di connessioni tra apparati di rete, scendiamo un ulteriore gradino ed esaminiamo la tecnologia che permette l'invio dei pacchetti lungo la connessione diretta tra di essi.

Per iniziare, definiamo qualunque apparato che gestisce un protocollo di livello 2 col nome di nodo e il canale di comunicazione che collega due nodi adiacenti col nome di connessione (link). Sono nodi, quindi, host, router, switch e punti di accesso via radio, e sono connessioni cavi, fibre ottiche e ponti radio che permettono la comunicazione tra i vari nodi.

Per capire meglio come il livello connessione si ponga rispetto al livello rete, consideriamo di nuovo un'analogia col mondo dei trasporti.

Un turista si reca in un'agenzia di viaggio e chiede di strutturare un viaggio da Milano a Londra. L'agente decide che il percorso più conveniente comprende il trasferimento in treno dal centro di Milano all'aeroporto di Malpensa, un volo diretto da Malpensa all'aeroporto londinese di Heathrow e, infine, il trasferimento in autobus da Heathrow al Victoria Air Terminal nel centro di Londra. Una volta che l'agenzia ha provveduto alla prenotazione delle tre tratte, il trasporto del turista da Milano a Malpensa diventa responsabilità delle ferrovie, il volo sino a Heathrow responsabilità della linea aerea e, infine, diventa responsabilità della compagnia di autobus il tragitto sino in centro a Londra.

Nella nostra analogia, il turista è un datagram, l'agenzia viaggi è un protocollo di routing

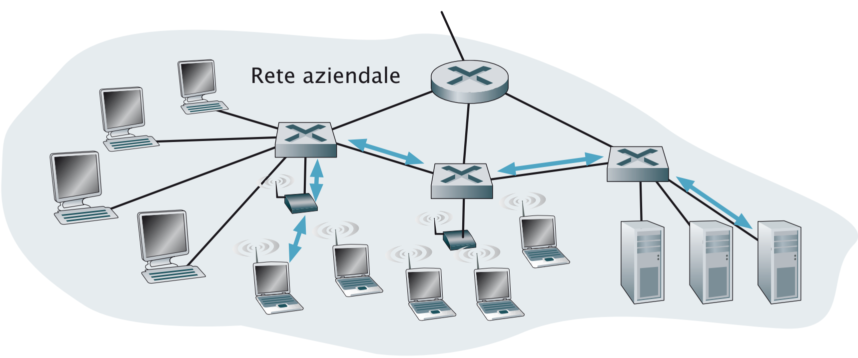


Figura **1**
Nodi a livello 2 in una rete locale

(quindi di livello 3), ogni posizione è un nodo, ognuna delle tratte è una connessione e le diverse modalità di trasporto sono i protocolli del livello connessione.

Notiamo come le tre tratte siano gestite da differenti compagnie di trasporto, le quali usano mezzi completamente diversi tra loro, ma forniscono tutte lo stesso servizio di base: la movimentazione dei passeggeri da una posizione alla posizione successiva.

Servizi di connessione • Benché il servizio fondamentale del livello connessione sia quello di muovere un datagram tra due nodi adiacenti, i diversi protocolli possono offrire ulteriori servizi, anche in considerazione del mezzo fisico su cui operano:

- *framing*: i protocolli del livello connessione incapsulano ogni datagram di livello rete in un frame, prima di trasmetterlo sulla connessione. Un frame consiste in un campo dati in cui è inserito il datagram, e in un certo numero di campi header. La struttura del frame è definita dal protocollo di livello connessione
- *accesso alla connessione*: esiste un protocollo di accesso al mezzo trasmissivo (MAC, *Medium Access Control*), che specifica le regole con cui un frame è trasmesso lungo la connessione. Nel caso di connessioni punto-punto con un solo mittente e un solo destinatario, il protocollo MAC è molto semplice, o addirittura inesistente: il mittente può inviare il frame in qualunque momento in cui la connessione è libera. Nel caso in cui più nodi condividono una sola connessione broadcast, il protocollo MAC serve a coordinare la trasmissione dei frame dai diversi nodi
- *consegna affidabile*: un protocollo di livello 2 fornisce un servizio di consegna affidabile quando garantisce di muovere ogni datagram da un nodo all'altro senza errori. Come per il servizio similare fornito dai protocolli di livello trasporto, questo servizio può essere ottenuto mediante meccanismi di riconoscimento e ritrasmissione. Il servizio di consegna affidabile è spesso utilizzato per connessioni che presentano intrinsecamente un alto tasso di errori, quali le trasmissioni via radio, mentre è considerato un doppione inutile dei protocolli di livello superiore quando vi sono connessioni a basso tasso di errori, quali il cavo in rame e la fibra ottica: per questa ragione, molti protocolli per tratte cablate non ne dispongono
- *rilevamento e correzione degli errori*: fenomeni di attenuazione del segnale e interferenza elettromagnetica lungo una tratta possono portare a una lettura sbagliata del frame da parte dell'hardware dei nodi, trasformando un 1 in uno 0 o viceversa. Poiché è uno spreco inoltrare un datagram errato, molti protocolli di livello connessione provvedono a rilevare l'errore e scartare il datagram. Per fare ciò, il protocollo prevede che il nodo mittente includa nell'header del frame dei bit di controllo, che sono poi usati dal nodo destinatario per verificare la correttezza del contenuto: questo controllo è più raffinato del semplice checksum utilizzato a livello 3 e 4 ed è implementato nell'hardware dei nodi. Alcuni protocolli prevedono anche la correzione degli errori, utilizzando nell'header meccanismi di controllo più raffinati, che consentono di determinare non solo la presenza dell'errore, ma anche il punto esatto dal frame in cui è avvenuto, permettendo quindi di intervenire senza necessità di ritrasmissione

Implementazione • Abbiamo visto nel capitolo precedente che il livello connessione, nei router, è gestito dalle porte di ingresso e di uscita: analizziamo, adesso, come è implementato all'interno di un host.

La maggior parte delle funzioni del livello connessione è contenuta in un hardware dedicato, chiamato *adattatore di rete* o *interfaccia di rete* (***NIC***, *Network Interface Card*), che gestisce direttamente framing, accesso e rilevamento degli errori; il chip che compie tutto ciò è chiamato ***controller***. Gli adattatori di rete cambiano struttura e funzioni secondo il mezzo cui si collegano: come vedremo più avanti, una scheda Ethernet e una WiFi utilizzano circuiti e protocolli molto diversi tra loro.

La figura 2 mostra la tipica struttura di un'interfaccia di rete all'interno di un host. Dal punto di vista fisico, essa non differisce da qualunque altra interfaccia di I/O, se non per il

particolare controller utilizzato.
La componente software del livello connessione, che gira sulla CPU dell'host, gestisce

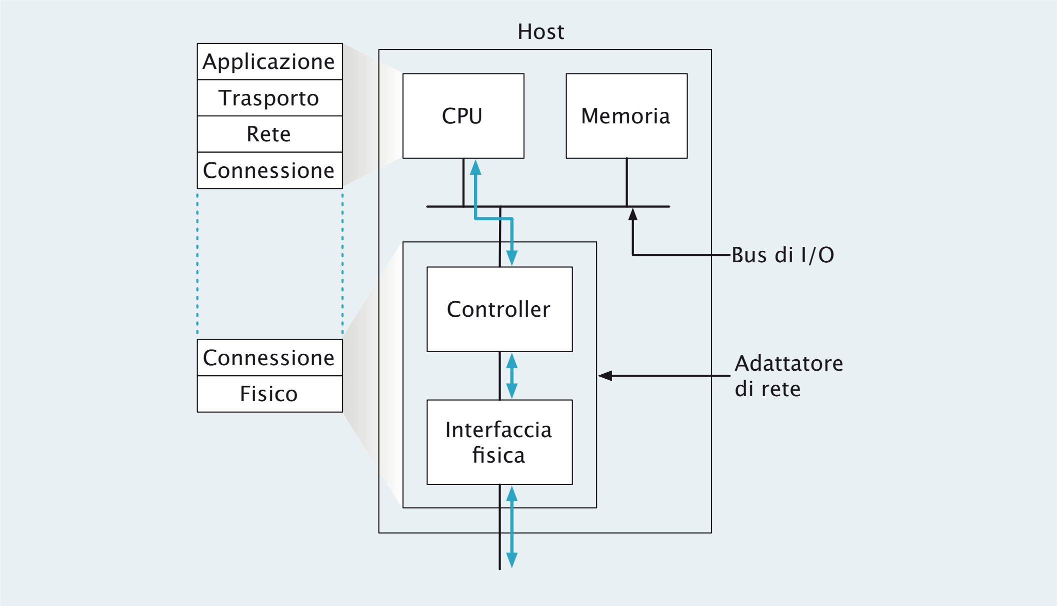


Figura **2**
Nodi a livello 2 in una rete locale

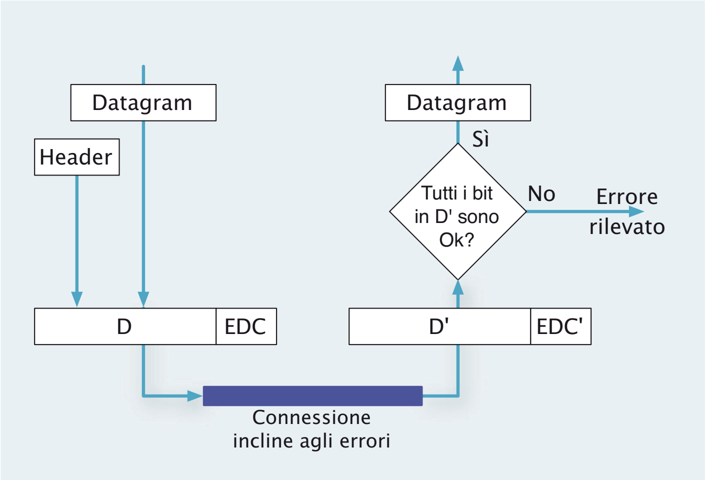
le funzioni di più alto grado, quali l'assemblaggio dell'indirizzamento, l'incapsulamento dei datagram e l'attivazione del controller. In ricezione, il software risponde agli interrupt del controller, gestisce le condizioni di errore e passa i datagram al livello superiore.

2 Rilevamento e correzione degli errori

Esistono molte tecniche di rilevamento e di correzione degli errori, alcune semplici, altre estremamente raffinate: la scelta dipende dal tipo di informazione che viene trasportata e dalla sua criticità rispetto alla tolleranza degli errori stessi.

Esaminando la figura 3, vediamo come la sfida per il destinatario sia quella di determinare se il datagram D' che ha ricevuto è identico all'originale inviato D, considerando che anche EDC' può essere stato soggetto a errori durante la trasmissione. La decisione che il destinatario prende ha comunque un certo grado di indeterminatezza, proprio a causa della possibilità che anche i bit di controllo siano stati modificati nel transito lungo la connessione: le tecniche di rilevamento e correzione permettono al destinatario di individuare spesso, ma non sempre, gli errori di trasmissione. Come conseguenza, il protocollo al livello 2 può passare un datagram scorretto al livello 3 o ignorare che il campo di controllo è stato corrotto.

Figura **3**
Rilevazione di un errore



Per minimizzare questo rischio, è necessario aumentare sempre più la complessità delle strategie di rilevamento, con conseguente appesantimento dei campi di controllo che il protocollo aggiunge al datagram all'interno del frame. In questa sede accenneremo alle tecniche più semplici e basilari, a partire dalle quali sono state sviluppate le strategie

di correzione più complesse e raffinate.

Controllo di parità • La forma più semplice di rilevamento degli errori è l'utilizzo di un singolo bit di parità. Supponiamo che l'informazione da trasmettere sia formata da una stringa di *d* bit, e aggiungiamo in fondo un ulteriore bit, il cui valore è tale che il numero di 1 nei $(d + 1)$ bit sia pari: se nella stringa abbiamo un numero dispari di 1, il bit di parità sarà 1, se ne abbiamo un numero pari, il bit di parità sarà 0. In questo modo, il destinatario deve solo contare il numero di 1 contenuto nella stringa di $(d + 1)$ bit: se sono dispari, allora c'è stato un errore, anche se è impossibile sapere quale sia il bit errato all'interno della stringa.

Se, però, nella trasmissione avvengono due errori, allora il bit di parità sembrerà corretto, e il protocollo non sarà in grado di riconoscere gli errori. Questa situazione è molto comune, in quanto gli errori sono normalmente causati da disturbi la cui durata è elevata rispetto alla velocità di trasmissione sulla connessione e, quindi, gli errori si presentano spesso raggruppati. In gergo informatico si dice che si clusterizzano (dall'inglese **cluster**, *grappolo*). In questi casi, la probabilità di errori non riconosciuti usando la tecnica del controllo di parità tende al 50%.

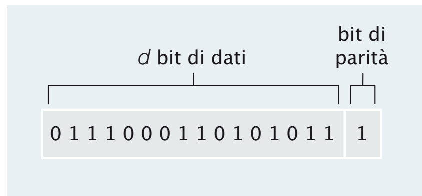


Figura **4**
Controllo di parità a 1 bit

La situazione migliora se utilizziamo una variante bidimensionale del controllo di parità nella quale i *d* bit della stringa D sono strutturati a matrice e divisi in *j* righe e *k* colonne, e in cui il bit di parità è calcolato per ogni riga e ogni colonna, in modo che il frame contenga $j + k + 1$ bit, l'ultimo dei quali è la parità delle parità.

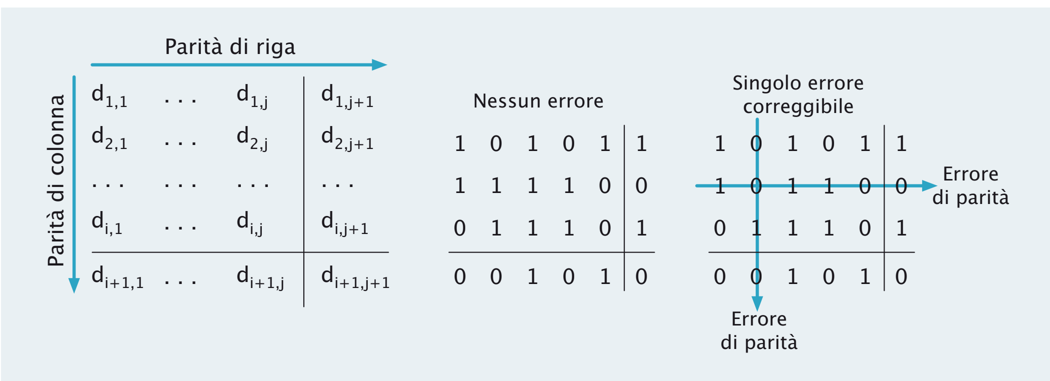


Figura **5**
Controllo di parità bidimensionale

Come vediamo in figura 5, utilizzando la parità bidimensionale, il protocollo può rilevare qual è il bit errato nella stringa e, quindi, procedere alla sua correzione. Allo stesso modo, l'errore di trasmissione di un bit di parità può essere rilevato e corretto, mentre una qualunque combinazione di due errori nella stringa può essere individuata, ma non corretta.

Checksum • La tecnica del ***checksum*** (letteralmente *somma di controllo*) consiste nel trattare la stringa costituita da *d* bit come una sequenza di numeri interi di *k* bit, con *k* sottomultiplo di *d*; la forma più semplice di checksum consiste nel calcolare la somma di tutti gli interi in cui è divisa la stringa e utilizzare il valore ottenuto come bit di rilevamento degli errori. Il checksum in Internet è basato su questo approccio: i byte di dati sono presi due a due, trattati come un intero a 16 bit e sommati. Il complemento a 1 della somma diventa il checksum e viene inserito nell'header del segmento. Il destinatario controlla il checksum prendendo il complemento a 1 della somma dei dati ricevuti, compreso il checksum stesso, e controllando che il risultato sia composto tutto da bit a1: se vi sono bit a 0, allora si è verificato un errore.

Il checksum richiede poche aggiunte ai pacchetti, quindi aumenta di poco il traffico di

rete e non appesantisce il lavoro di elaborazione software da parte degli host e dei router: per questo è utilizzato dai protocolli di trasporto, quali TCP e UDP, ma garantisce una protezione piuttosto debole e non viene quasi mai utilizzato a livello connessione.

A5-01
Cyclic Redundancy Check

Controllo a ridondanza ciclica • Questo metodo, detto normalmente ***CRC*** (*Cyclic Redundancy Check*), è attualmente il più usato per il rilevamento e la correzione degli errori nelle reti. I codici CRC sono noti anche come *codici polinomiali*, dato che è possibile considerare la stringa come un polinomio i cui coefficienti sono i valori 0 e 1 dei bit nella stringa.

3 Connessioni ad accesso multiplo

Abbiamo visto nell'introduzione che esistono due tipi di connessione di rete: quelle *punto-punto*, che consistono in un solo mittente a un capo della connessione e un solo destinatario all'altro capo, e quelle *broadcast*, che possono avere più nodi mittenti e destinatari, tutti collegati allo stesso canale. In queste ultime connessioni, si pone il problema di coordinare l'accesso di molti nodi a un unico canale condiviso.

Broadcast • Il concetto di broadcast ci è familiare, dato che questo tipo di trasmissione è usato dalla radio e dalla televisione. Esiste, tuttavia, una differenza fondamentale tra questi sistemi e le reti: nel caso della radio e della televisione si tratta di trasmissioni unidirezionali, mentre in una rete i nodi possono sia ricevere che trasmettere. Quindi è necessario stabilire un insieme di regole sull'occupazione del mezzo trasmissivo da parte dei nodi stessi: in caso contrario, tutti cercherebbero di inserirsi contemporaneamente, e si assisterebbe alla congestione del canale.

Volendo fare un'analogia, possiamo considerare una classe scolastica, in cui docente e allievi condividono lo stesso mezzo trasmissivo e, quindi, devono sottostare ad alcune regole fondamentali per evitare la più totale confusione:

- dare a tutti la possibilità di parlare
- non parlare mentre un altro sta parlando
- non monopolizzare la conversazione
- alzare la mano quando si vuole porre una domanda
- non interrompere se qualcuno sta parlando
- non addormentarsi mentre qualcuno sta parlando

Come si vede, abbiamo generato un protocollo. Le reti, similmente, necessitano di protocolli che gestiscano l'accesso di nodi multipli allo stesso canale, e che vengono detti per questo *protocolli ad accesso multiplo*.

Protocolli ad accesso multiplo • Poiché tutti i nodi sono in grado di trasmettere frame, può capitare che due (o più) di essi li trasmettano nello stesso istante: quando ciò accade, tutti i nodi ricevono tutti i frame contemporaneamente, e di conseguenza i frame collidono tra loro alla porta di ingresso di tutti i nodi. In questa situazione, i nodi non sono in grado di distinguere quali bit sono contenuti in un frame, e in quale tra i tanti: quindi non riescono a estrarre alcuna informazione da quello che ricevono: tutti i frame coinvolti nella collisione, perciò, sono perduti e il canale di trasmissione è inutilizzabile durante tutta la permanenza della collisione.

Al fine di assicurare che il canale broadcast funzioni correttamente quando sono attivi più nodi, è necessario coordinarne in qualche modo le trasmissioni: questo coordinamento è il compito dei protocolli ad accesso multiplo.

Le decine di protocolli ad accesso multiplo utilizzati nel tempo possono essere classificati in tre sole categorie: protocolli ***a partizione di canale*** (*Channel Partitioning Protocols*), protocolli ***ad accesso casuale*** (*Random Access Protocols*) e protocolli ***a turno*** (*Taking Turns Protocols*).

La trattazione di questi protocolli è complessa, sia dal punto di vista strutturale, sia da quello matematico. Proponiamo online la descrizione del protocollo ***CSMA/CD*** (*Carrier Sense Multiple Access/Collision Detection*, ovvero *accesso multiplo con rilevamento della portante/rivelazione delle collisioni*), utilizzato nello standard Ethernet, del quale parleremo nel prossimo paragrafo.

A5-02
Il protocollo CSMA/CD

4 LAN commutate

Host e router hanno, oltre agli indirizzi di livello rete, anche indirizzi di livello connessione. In realtà, questi ultimi sono associati direttamente alle interfacce di rete: un host o un router dotati di più NIC, quindi, possiedono altrettanti indirizzi di livello 2. Uno switch di livello 2, invece, non possiede alcun indirizzo associato alle proprie porte, in quanto opera la commutazione in modo trasparente, quindi senza richiedere che il mittente indirizzi esplicitamente i frame sugli switch interposti sulla strada verso il destinatario.

Gli indirizzi di livello connessione vengono normalmente chiamati MAC address e, in tutte le architetture LAN attualmente usate, sono formati da 6 byte, ciascuno dei quali è espresso in notazione esadecimale, per un totale di 248 indirizzi possibili. I MAC address sono assegnati in modo permanente alla NIC al momento della sua produzione, quindi sono indissolubilmente legati all'hardware. Poiché le NIC sono prodotte da molte aziende, che sono distribuite in tutto il mondo, è necessario che esista un'organizzazione che gestisce l'assegnazione degli indirizzi: questo compito è affidato alla ***IEEE*** (si legge *I triple E*). Quando un produttore vuole immettere nuove NIC sul mercato, acquista dalla IEEE un blocco di indirizzi, che vengono forniti con i primi 3 byte fissi, lasciando al produttore la possibilità di creare combinazioni qualsiasi, purché uniche, con i restanti 3 byte: ciò significa che ogni produttore ha a disposizione 224 (ossia 16.777.216) indirizzi diversi. La IEEE, a sua volta, ha a disposizione un ugual numero di blocchi di indirizzi.

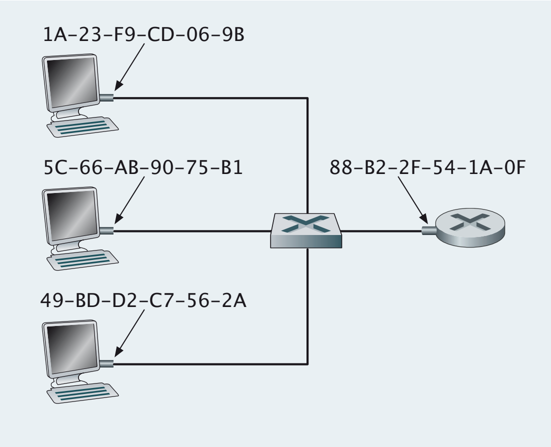


Figura **6**
Associazione tra NIC e MAC address

Gli indirizzi MAC hanno una struttura piatta e non cambiano al cambiare della rete cui è connessa la NIC (ricordiamo che, invece, gli indirizzi IP hanno struttura gerarchica e cambiano ogni volta che l'host si connette a una rete diversa). Volendo fare un'analogia, il MAC address corrisponde al codice fiscale, che ha struttura piatta e segue le persone ovunque, mentre l'indirizzo IP corrisponde all'indirizzo postale, che è gerarchico e deve essere cambiato ogni volta che una persona trasloca.

Quando una NIC vuole inviare un frame a uno specifico destinatario, inserisce nel frame il MAC address relativo e lo invia sulla LAN; a volte, però, uno switch decide di effettuare un broadcast del frame su tutte le porte, quindi alcuni host ricevono un frame non destinato a loro. Ogni NIC, quindi, deve verificare che l'indirizzo contenuto nei frame che riceve corrisponda al proprio: se non corrisponde, si limita a scartarlo, mentre se corrisponde estrae il datagram e lo passa al livello superiore.

Può accadere che un host voglia che tutti gli altri host nella LAN ricevano e processino il frame che sta per inviare: in questo caso, viene inserito un particolare MAC address, formato

da una sequenza di 1 (in esadecimale FF-FF-FF-FF-FF-FF), che viene accettato da tutte le NIC.

ARP • Abbiamo visto che per indirizzare un datagram sono necessari sia l'indirizzo IP che il MAC address: è necessario, quindi, avere a disposizione un protocollo che provveda alla traduzione di uno nell'altro. Nell'Internet, e nella maggior parte delle LAN, questo lavoro è fatto dall'***ARP*** (*Address Resolution Protocol*).

Partendo dalla figura 7, supponiamo che l'host con indirizzo IP 222.222.222.220 voglia inviare un datagram all'host con indirizzo 222.222.222.222.

Per poterlo fare, l'host deve passare alla propria NIC non solo il datagram, ma anche il MAC address corrispondente all'indirizzo IP del destinatario. La NIC del mittente, quindi, costruisce un frame che contiene il datagram e il MAC address del destinatario, dopo di che lo invia sulla rete: nel nostro esempio, la NIC fornisce al proprio modulo ARP l'indirizzo IP 222.222.222.222 e ne riceve il corrispondente MAC address, 49-BD-D2-C7-56-2A.

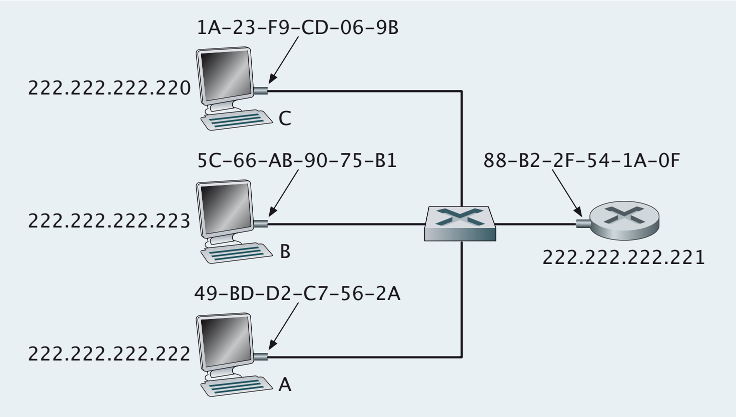


Figura **7**
MAC address e indirizzi IP

Vediamo, quindi, che ARP risolve l'indirizzo IP nel MAC address, analogamente al DNS, che risolve i nomi di dominio in indirizzi IP. La differenza fondamentale sta nell'ampiezza della capacità di risoluzione: DNS risolve i nomi per host che si trovano in qualunque punto della rete, mentre ARP funziona solo all'interno di una sottorete.

Tabella ARP • Per effettuare la traduzione tra gli indirizzi, gli host e i router possiedono ognuno una tabella di mappatura (vedi tabella 1), composta da tre colonne. La terza colonna indica il tempo dopo il quale il record verrà eliminato dalla tabella, in modo da non intasarla con registrazioni ormai inutili.

Tabella **1**

Indirizzo IP	MAC address	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Questa tabella non contiene necessariamente tutti i MAC address presenti nella sottorete, ma solo quelli con cui l'host si è messo in comunicazione.

Per generare i record della tabella, l'host costruisce un pacchetto particolare, chiamato *pacchetto ARP*, che ha lo scopo di chiedere a tutti gli host e router della sottorete di determinare il MAC address corrispondente all'indirizzo IP cui l'host è interessato.

Torniamo al nostro esempio, in cui l'host 222.222.222.220 vuole inviare un datagram all'host 222.222.222.222: vediamo dalla tabella 1 che la NIC non conosce il MAC address corrispondente. Il pacchetto ARP che il nostro host invia contiene il proprio indirizzo IP, il proprio MAC address, l'indirizzo IP del destinatario e il MAC address broadcast:

222.222.222.220
1A-23-F9-CD-06-9B
222.222.222.222
FF-FF-FF-FF-FF-FF

Il pacchetto ARP è, quindi, ricevuto da tutte le NIC della sottorete, ma solo quella corrispondente al destinatario risponde, inviando al mittente un pacchetto ARP di risposta, contenente il proprio MAC address, che viene utilizzato per generare il nuovo record nella tabella.

Notiamo che la struttura del pacchetto ARP è asimmetrica, in quanto la richiesta è inviata in broadcast, mentre la risposta è mirata al MAC address del richiedente.

La tabella ARP, inoltre, è di tipo *plug-and-play*, in quanto non richiede intervento da parte di un amministratore di rete, ma si forma automaticamente quando vi è una richiesta di connessione nella sottorete: quando un host viene sconnesso dalla rete, dopo un certo tempo il record relativo viene cancellato.

Indirizzamento esterno • Quanto detto sinora descrive la connessione a livello 2 all'interno di una sottorete. La situazione diviene più complessa quando un host vuole inviare un datagram a un destinatario che si trovi all'esterno della propria sottorete.

Nella figura 8 troviamo due semplici sottoreti, collegate tra loro da un router.

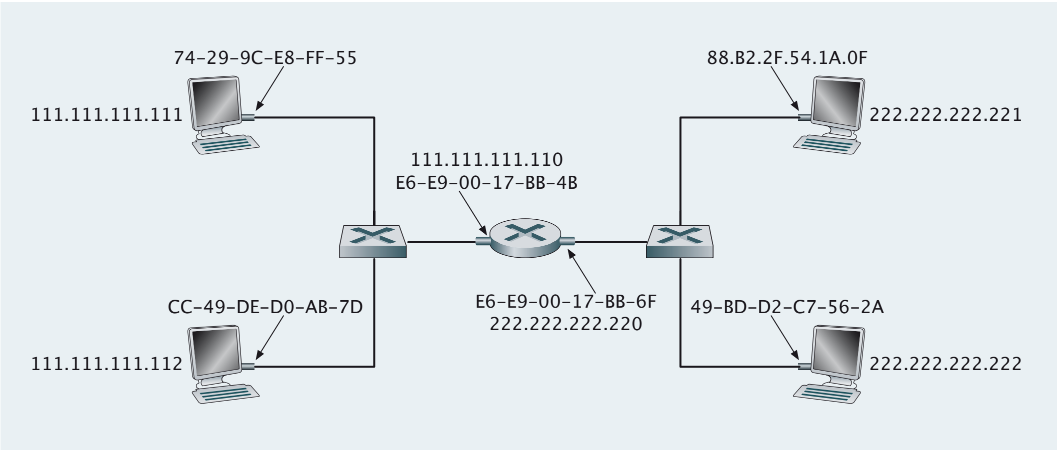


Figura **8**
Due sottoreti connesse attraverso un router

Notiamo come ogni host abbia un adattatore di rete con relativo MAC address e un proprio indirizzo IP, mentre il router ha due interfacce, quindi due indirizzi IP e due porte, ciascuna delle quali ha la propria NIC, il proprio MAC address e il proprio modulo ARP. La sottorete 1 ha indirizzo 111.111.111/24 e la sottorete 2 ha indirizzo 222.222.222/24.

Supponiamo che l'host 111.111.111.111 voglia inviare un datagram a 222.222.222.222.

Il mittente passa il datagram con l'indirizzo IP del destinatario alla propria NIC, ma questa non conosce il MAC address del destinatario (e non può conoscerlo, perché si trova fuori dalla sottorete). Quindi ARP non è in grado di risolverlo.

Poiché il nodo di scambio tra le due sottoreti è il router, il mittente deve usare come destinazione il MAC address della porta del router connessa alla propria sottorete: se questa non è presente nella tabella ARP, invia un pacchetto ARP broadcast come descritto sopra.

Il router legge l'indirizzo IP e lo trova nella propria tabella di routing, quindi risponde al mittente inviando il proprio MAC address, ossia E6-E9-00-17-BB-4B.

A questo punto, l'host invia il frame al router, la cui interfaccia lo passa al livello rete, che provvedere a instradare il datagram sulla porta connessa alla sottorete 2, dove risiede l'host con l'indirizzo IP indicato come destinatario.

La NIC della porta di uscita invia un pacchetto ARP con l'indirizzo IP del destinatario, che risponde con un pacchetto ARP contenente il proprio MAC address, ossia 49-BD-D2-C7-56-2A. A questo punto, il router può generare il frame in uscita e inviarlo al destinatario.

Ethernet • Negli anni '80 e nei primi anni '90 esistevano molte architetture che si contendevano il mercato delle LAN: prime fra tutte FDDI, Token Ring, ATM ed Ethernet. Oggi le prime due sono scomparse, e quello che rimane di ATM è migrato sulle reti geografiche, in particolare come base del DSL.

Le ragioni per cui Ethernet ha conquistato completamente il mercato sono molte e profondamente correlate, ma possiamo semplificare dicendo che è risultata vincente per la maggiore semplicità di realizzazione e la continua introduzione di versioni con una maggior velocità di trasmissione, a fronte di alcuni difetti insiti nella struttura, che però non sono risultati importanti nella implementazione pratica dell'architettura.

L'introduzione all'inizio degli anni '90 degli switch e dei cablaggi strutturati con configurazione a stella, oltre al totale ed esclusivo supporto che il produttore Cisco ha offerto a Ethernet, ha definitivamente eliminato la concorrenza. La semplicità intrinseca del sistema e la grande diffusione hanno, infine, portato al crollo dei prezzi dell'hardware, rendendo Ethernet la tecnologia di gran lunga più economica per la realizzazione delle LAN.

Notiamo come il termine Ethernet indichi, in realtà, una famiglia di tecnologie per reti locali, i cui standard ne definiscono le specifiche tecniche sia a livello fisico (connettori, cavi, specifiche di trasmissione) sia a livello connessione.

Struttura del frame • Un frame Ethernet è formato da sei campi, come mostrato in figura 9.

Figura **9**

Preambolo	Indirizzo destinazione	Indirizzo mittente	Tipo	Dati		CRC

Eccone la descrizione:

- *Preambolo*: è il campo iniziale, e ha lunghezza 8 byte. I primi 7 byte sono uguali, con valore 10101010, mentre l'ultimo ha valore 10101011: il loro compito è di svegliare la NIC ricevente e sincronizzarne il clock con quello della NIC mittente; gli ultimi due bit dell'ottavo byte, che sono gli unici 1 consecutivi, comunicano al destinatario che i byte seguenti contengono informazioni. La sincronizzazione dei clock si rende necessaria perché la NIC trasmittente non riesce mai a trasmettere alla velocità nominale, ma presenta una deriva dovuta alle tolleranze di produzione, che è a priori ignota alle altre NIC
- *Indirizzo di destinazione*: contiene il MAC address del destinatario, quindi ha lunghezza 6 byte: quando una NIC riceve un frame in cui il contenuto di questo campo corrisponde al proprio MAC address o al broadcast, passa il contenuto del campo dati al livello superiore; in caso contrario, scarta il frame
- *Indirizzo sorgente*: contiene il MAC address del mittente, quindi ha anch'esso lunghezza 6 byte, e viene usato da ARP per generare i record nella tabella di mappatura
- *Tipo*: questo campo, di lunghezza 2 byte, indica il protocollo utilizzato dal livello rete. Un host, infatti, può usare protocolli di rete diversi da IP, quali Novell IPX o AppleTalk (anche se ormai tutti convergono su IP), ognuno dei quali ha un proprio numero di tipo standardizzato; anche il pacchetto ARP ha un proprio numero di tipo. Ethernet deve sapere cosa è usato nel datagram, per poterlo passare al giusto protocollo a livello superiore. Notiamo come il campo *Tipo* sia analogo al campo *Protocollo* all'interno del datagram a livello rete e al campo *Porta* nel segmento a livello trasporto: tutti servono ad allacciare un protocollo su un livello a un altro protocollo al livello superiore
- *Dati*: contiene il datagram IP o di altri protocolli, indicati dal campo *Tipo*. Poiché la capacità massima del campo è 1.500 byte, un datagram di dimensioni superiori deve essere frammentato, come abbiamo visto nel capitolo precedente. Questo campo ha anche una dimensione minima di 46 byte per cui, se il datagram è più breve, il campo deve essere riempito con bit privi di significato. Quando il datagram viene ricevuto e passato al livello superiore, il protocollo utilizza il campo Lunghezza presente nell'header del datagram per rimuovere i bit inutili
- *CRC*: come abbiamo visto nel paragrafo 2, questo campo, di lunghezza 4 byte, consente alla

NIC del destinatario di effettuare il rilevamento e la correzione degli errori

Tecnologia • Quando Ethernet è nata, oltre trent'anni fa, utilizzava una tecnologia molto diversa dall'attuale: adottava un cavo coassiale che univa tutti i nodi in serie, formando quello che tecnicamente è detto un bus.

Quando dovevano definire un'unica sottorete, i vari bus erano collegati tra loro da dispositivi chiamati hub, il cui funzionamento era estremamente primitivo. Un hub, infatti, si limitava a comportarsi come un amplificatore, e ritrasmetteva su tutte le uscite, senza alcun controllo, ogni singolo bit che si presentava a un ingresso, fungendo semplicemente da amplificatore.

È chiaro che una simile topologia è totalmente broadcast e richiede protocolli e strumenti per gestire le singole comunicazioni all'interno del mezzo trasmissivo, impedendo o riducendo al massimo i problemi di collisione e di congestione della rete: in questo il protocollo MAC dell'Ethernet si è rivelato in grado di gestire reti anche di una certa complessità, purché strutturate in modo tale che tra due nodi vi fosse un solo percorso possibile.

Oggi Ethernet è implementata con architetture a stella multipla, ciascuna con centro su uno switch, a sua volta connesso a stella con altri switch, oppure collegato tramite uplink a uno o più router che forniscono la connessione verso l'esterno della sottorete.

Le varie versioni dell'Ethernet sono definite dallo standard IEEE 802.3 e, negli anni, hanno cambiato radicalmente specifiche.

Il nome di tutte le versioni è, comunque, definito univocamente: un numero, che definisce la velocità della connessione, la parole BASE, che indica come il mezzo fisico porti solo banda base Ethernet, seguita da un trattino e una o più lettere o cifre che corrispondono a uno specifico mezzo trasmissivo. I vecchi standard 10BASE-2 e 10BASE-5 indicavano l'utilizzo di due tipi di cavi coassiali con caratteristiche diverse e velocità 10 Mbps; nel tempo sono nate le versioni per architettura a stella, quindi con link punto-punto, a velocità crescente: 10BASE-T, 100BASE-T, 1000BASE-TX, 1000BASE-FX, 10GBASE-X, che vanno da 10 Mbps a 10 Gbps.

A5-03
Lo standard IEEE 802.3

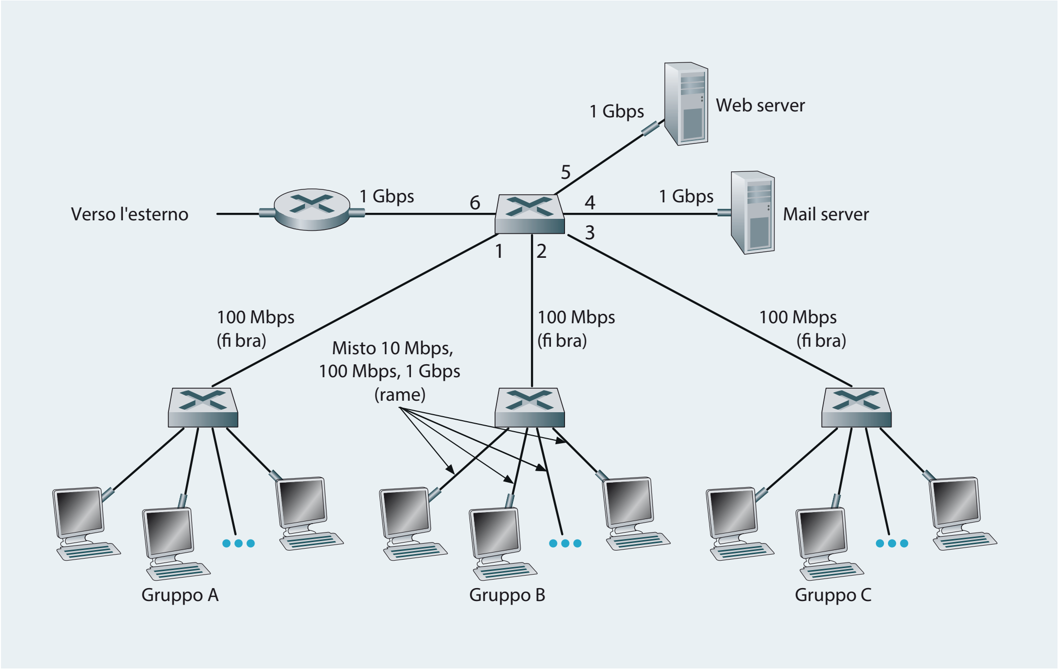


Figura **10**
Struttura di una LAN gerarchica

VLAN • Le moderne reti locali di medie e grandi dimensioni sono configurate in modo gerarchico, in cui ogni gruppo è dotato di una propria rete commutata, connessa alle altre

reti commutate attraverso una struttura di switch ad albero (vedi figura 10).

Questa configurazione, che apparentemente può funzionare perfettamente, presenta invece tre inconvenienti principali:

- *mancanza di isolamento del traffico*: benché la gerarchia localizzi il traffico interno di ogni gruppo, il traffico broadcast (frame con messaggi ARP e DHCP o con destinazioni non ancora apprese dagli switch) deve ancora attraversare l'intera rete: ciò non è auspicabile, sia per ragioni di degrado delle prestazioni della rete, sia perché si riduce grandemente il livello di sicurezza delle informazioni in transito. La soluzione più semplice è, ovviamente, quella di interporre uno o più router tra i vari gruppi, ma anche questa soluzione degrada le prestazioni globali della rete, dato che richiede di risalire al livello 3
- *uso inefficiente degli switch*: più è alto il numero dei gruppi nella rete, più sono gli switch di livello inferiore che, però, difficilmente hanno tutte le porte occupate e gestiscono un traffico molto inferiore alle proprie possibilità
- *gestione degli utenti*: se un utente si sposta da un gruppo all'altro, il cablaggio fisico deve essere modificato per cambiare lo switch cui l'utente è connesso. Se, poi, esistono utenti che appartengono a più gruppi, il problema diventa ancora più complesso

Tutte queste difficoltà possono essere superate generando delle ***VLAN***, ossia delle *reti locali virtuali*, in cui l'appartenenza di un utente a un gruppo è definita indipendentemente dalla sua posizione fisica nella rete, almeno entro certi limiti.

Per fare ciò, è necessario utilizzare switch che supportino la creazione di VLAN, ossia che permettano di definire un certo numero di reti virtuali, separate tra loro, all'interno di una singola rete fisica.

Questi switch sono dotati di un software che permette di assegnare le singole porte a uno di più gruppi, indipendentemente dalla posizione fisica nello switch stesso, e generare per ogni gruppo un dominio di broadcast separato e, quindi, isolarlo dagli altri gruppi. In questo modo, un unico switch può assolvere alle stesse funzioni di due o più switch che non supportano le VLAN.

A questo punto, però, i gruppi sono totalmente isolati tra loro. Se il router ha un certo numero di porte inutilizzate e solo due o tre gruppi, basta assegnare una o due di queste porte a ogni gruppo e connetterle direttamente tra loro. Si tratta, però, di una soluzione non scalabile e poco efficiente: l'ideale sarebbe effettuare un routing locale tra i gruppi. Questo è lo scopo per cui gli switch che supportano le VLAN hanno anche un router interno, che consente di evitare un router esterno e l'occupazione inutile di porte.

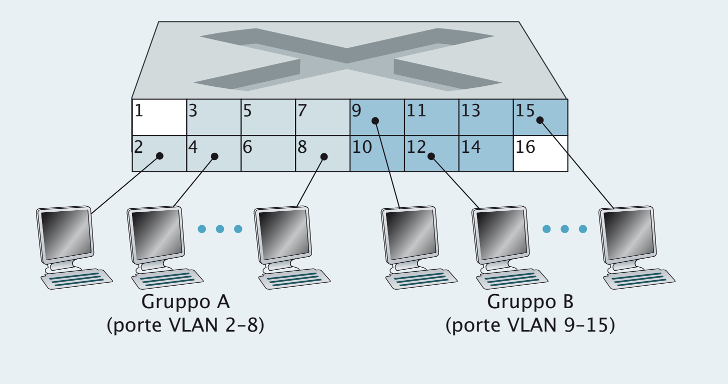


Figura **11**
Switch con due VLAN configurate

Trunking • In una topologia di rete complessa, non è possibile identificare sempre i gruppi con uno specifico switch: ogni VLAN deve possedere porte su un certo numero di switch, per ovvie ragioni di scalabilità e di flessibilità. Per ottenere una simile configurazione esistono due possibilità, la più semplice delle quali è riservare una porta per ogni gruppo su entrambi gli switch e interconnetterle fisicamente, come si vede in figura 12A.

Una soluzione più raffinata è utilizzare uno switch che possiede una porta speciale, detta porta di trunking, che serve a interconnettere tra loro i vari switch VLAN (figura 12B). Questa porta appartiene a tutte le VLAN e tutti i frame inviati da ogni gruppo vengono inviati attraverso di essa all'altro switch che, però, è in grado di discriminare da quale VLAN

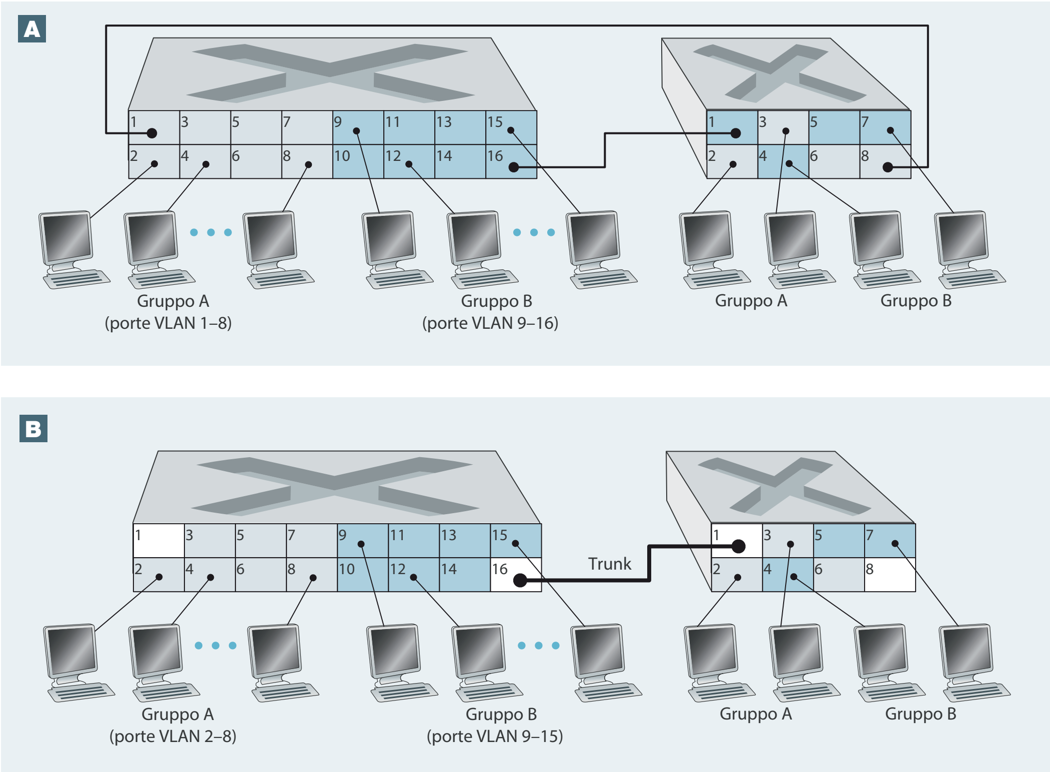


Figura **12**
VLAN distribuite
a) porte riservate
b) trunking

proviene un frame e, quindi, di instradarlo solo all'interno del gruppo relativo.

Per distinguere a quale VLAN appartiene un frame, è stato definito un frame Ethernet esteso, dotato di una struttura standard a cui sono aggiunti due campi in quarta e quinta posizione (figura 13).

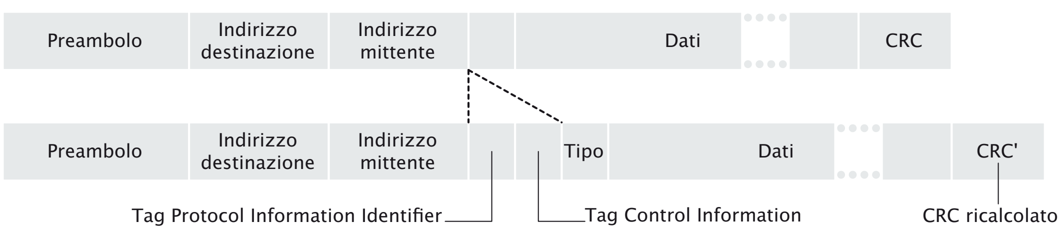


Figura **13**

Il frame esteso è così strutturato:

- Preambolo
- Indirizzo di destinazione
- Indirizzo sorgente
- ***TPID*** (*Tag Protocol Information Identifier*). Questo campo ha lunghezza 2 byte e contiene sempre il valore esadecimale 81-00: serve a dichiarare che il frame è di tipo esteso
- ***TCI*** (*Tag Control Information*). Contiene un identificatore della VLAN di lunghezza 12 bit, più tre bit la cui funzione è simile al campo TOS nel datagram IP
- Tipo
- Dati
- CRC

Le VLAN che abbiamo descritto si basano sulla formazione di gruppi di porte negli switch, ma è possibile generare VLAN anche sulla base dei MAC address degli host e su protocolli di livello rete, quali IPv4, IPv6 o AppleTalk.

5 Datacenter

Negli ultimi anni, grandi società che operano su Internet, quali ad esempio Amazon, Apple, Facebook, Google, Microsoft, hanno costruito datacenter giganteschi, ognuno contenente decine o centinaia di migliaia di server, che supportano molte diverse applicazioni.

Altri grandi datacenter sono stati realizzati da multinazionali che necessitano di enormi potenze di calcolo per la loro ricerca, quali case automobilistiche, società petrolifere, produttori e distributori di energia. Da ultimo, alcuni ISP forniscono anche un servizio di housing e hosting per aziende di dimensioni minori, ma che necessitano comunque di centinaia o migliaia di server e che non ritengono vantaggioso realizzare un datacenter interno.

Ovviamente, la densità degli host e la mole dei dati che questi si scambiano tra loro e con il mondo esterno richiedono una rete con caratteristiche particolari, che riduca al minimo i colli di bottiglia: l'efficienza computazionale di un datacenter si misura proprio con la percentuale di utilizzo delle CPU dei server.

Gli host possono fare lavori molto diversi: fornire contenuti quali pagine Web e streaming audio/video, gestire email e grandi basi dati, eseguire grandi moli di calcoli in parallelo.

Questi server sono generalmente prodotti commerciali, comprendenti la CPU, memoria e dispositivi di archiviazione su disco, anche se alcuni grandi utilizzatori, ad esempio Google, hanno finito per progettare e realizzare le proprie macchine, utilizzando architetture ottimizzate per il particolare utilizzo che ne viene fatto.

Gli host sono collocati ad alta densità in armadi rack, in numero variabile tra 20 e 40, secondo che si tratti di unità rack 19" o di schede blade inserite in sottotelai dedicati. Gli armadi contengono anche la distribuzione dell'alimentazione e sono costruiti in modo da massimizzare la rimozione del calore sviluppato.

Nella parte superiore di ogni rack c'è uno switch, chiamato per questo ***TOR*** (*Top Of the Rack*), che interconnette gli host tra loro e con gli altri switch del datacenter. Più specificamente, ogni host nel rack ha un'interfaccia di rete collegata allo switch TOR, che possiede anche alcune ulteriori porte che possono essere utilizzate per la connessione ad altri switch: in genere, queste porte vengono chiamate ***uplink*** e hanno caratteristiche di banda superiori di un ordine di grandezza rispetto alle altre porte. Inoltre, sono dotati di connessioni particolari con lo switching fabric interno.

Nei datacenter di alto livello la business continuity è imperativa. Per questo, gli host hanno sempre due interfacce di rete, che si collegano a due switch TOR messi in parallelo: in questo modo, sia che si guasti un'interfaccia di rete del server, sia che smetta di funzionare uno switch TOR, non vi sarà interruzione del servizio.

La rete di un datacenter supporta due tipi di traffico: quello che arriva da client esterni e quello tra gli host interni. Per gestire i flussi tra i client esterni e i server interni, la rete del datacenter comprende anche un certo numero di *router di confine* (*Border Router*), che collegano la rete del datacenter all'Internet: la rete di un datacenter, quindi, interconnette i rack tra di loro e con i border router.

Load balance • Nei datacenter che forniscono contenuti sull'Internet, la rete interna deve supportare la grande quantità di richieste provenienti dall'esterno, e che si rivolgono ad applicazioni diverse.

Per gestire in modo ottimale questa situazione, ogni applicazione è associata a un indirizzo IP pubblico, al quale i client inviano le loro richieste e dal quale ricevono le risposte.

All'interno del datacenter, le richieste esterne sono dirette a un ***load balancer***, letteralmente *bilanciatore di carico*, il cui lavoro consiste nel distribuirle tra i vari host che gestiscono la specifica applicazione, bilanciando il carico tra di essi in funzione del loro tasso di occupazione istantanea: in questo modo, si evita di sovraccaricare alcuni server e lasciarne inattivi altri,

e si distribuisce uniformemente il traffico tra le connessioni della rete.

A volte i load balancer sono chiamati switch di livello 4, dato che prendono decisioni sulla base, oltre che dell'indirizzo IP, della porta di destinazione del pacchetto da instradare che, come abbiamo visto, si trova al livello trasporto.

Quando riceve una richiesta per una specifica applicazione, il load balancer la inoltra a uno dei server a essa dedicati, che a sua volta può invocare i servizi di altri server per aiutare a processarla; quando il server ha elaborato la risposta, la invia a sua volta al load balancer, che a sua volta la gira al client esterno.

Il load balancer, quindi, oltre a provvedere alla distribuzione delle richieste sulla base dell'applicazione e del carico, compie anche una funzione NAT, traducendo l'indirizzo IP pubblico in quello interno del server scelto, e operando all'inverso quando i server inviano messaggi all'esterno.

Come nei router NAT, la scelta del percorso avviene sulla base della porta indicata nei pacchetti in transito. In questo modo i client esterni non sono in grado di contattare direttamente i server, e la struttura delle rete locale rimane invisibile.

Architettura gerarchica • Un datacenter che contiene al massimo qualche centinaio di server può essere gestito efficacemente da una LAN molto semplice, consistente in un border router, un load balancer e poche decine di rack, ciascuno con il proprio switch TOR. Per sistemi di decine o centinaia di migliaia di host, invece, è necessario impiegare una struttura gerarchica, come quella rappresentata in figura 14.

Al vertice della piramide, il border router è connesso ai router di accesso, che possono essere presenti in numero elevato. Sotto di essi si trovano tre file di switch. Ogni router di accesso è collegato a un certo numero di switch della fila superiore.

Ogni switch della fila superiore è collegato a un load balancer e a vari switch della fila intermedia, e ogni switch della fila intermedia è a sua volta collegato agli switch TOR di una parte degli armadi.

Tutte le connessioni utilizzano tipicamente Ethernet sia per il livello di connessione che per quello fisico, con un mix di tratte in rame e in fibra, secondo le lunghezze e le larghezze di banda necessarie, come vedremo nel prossimo capitolo.

Osserviamo come gli host sotto ogni router di accesso formino una singola sottorete, separata dalle altre: allo scopo di localizzare il più possibile il traffico ARP, che è broadcast, ognuna di queste sottoreti è suddivisa in un certo numero di VLAN, ciascuna comprendente al massimo qualche centinaio di host.

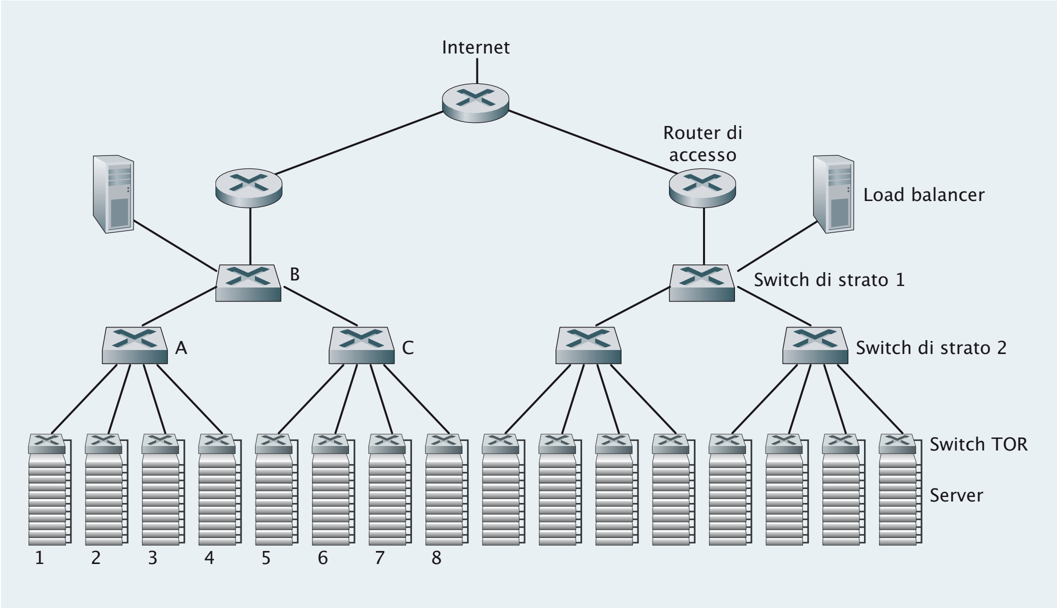


Figura **14**
Architettura tipica di un datacenter

Benché l'architettura gerarchica convenzionale risolva i problemi di scala dei datacenter, soffre di una limitata capacità di connessione tra gli host delle diverse sottoreti.

Considerando ancora la figura 14, supponiamo che ogni host sia connesso al proprio switch TOR con un'Ethernet a 1 Gbps, e che le connessioni tra gli switch avvengano a 10 Gbps. In questo modo, due server nello stesso armadio possono sempre comunicare a 1 Gbps ma, se vi sono molti flussi simultanei all'interno della rete del datacenter, la comunicazione tra host di armadi diversi può risultare molto più lenta, malgrado la maggiore larghezza di banda disponibile.

Poniamo, per semplicità, che siano presenti 40 flussi contemporanei: 10 tra gli armadi 1 e 5, 10 tra gli armadi 2 e 6, 10 tra gli armadi 3 e 7, e 10 tra gli armadi 4 e 8. Se ogni flusso condivide in modo paritetico con gli altri flussi la capacità delle connessioni che attraversa allora la tratta A - B di capacità 10 Gbps è attraversata contemporaneamente da 40 flussi, che dispongono ognuno di 250 Mbps, ossia un quarto delle possibilità delle interfacce di rete dei server.

Architettura a connessione totale • La soluzione più semplice per superare la limitazione descritta sopra sarebbe utilizzare tratte con larghezza di banda superiore, ma esiste un limite alle disponibilità tecnologiche e, comunque, i costi crescerebbero esponenzialmente.

Un'altra soluzione apparentemente facile sarebbe quella di riunire tutti i server che lavorano per la stessa applicazione negli stessi armadi, all'interno di una sola sottorete: questa topologia, però, finirebbe per rendere molto rigida l'allocazione dei server nel datacenter, contraddicendo una delle sue necessità fondamentali, ossia la flessibilità di alloggiamento dei server, condizione che permette di utilizzare al massimo la disponibilità di spazio degli armadi, indipendentemente dalla loro posizione fisica.

La principale tendenza attuale per superare le limitazioni che abbiamo descritto consiste nella sostituire la struttura gerarchica con una *topologia a connessione totale*, come quella riprodotta in figura 15.

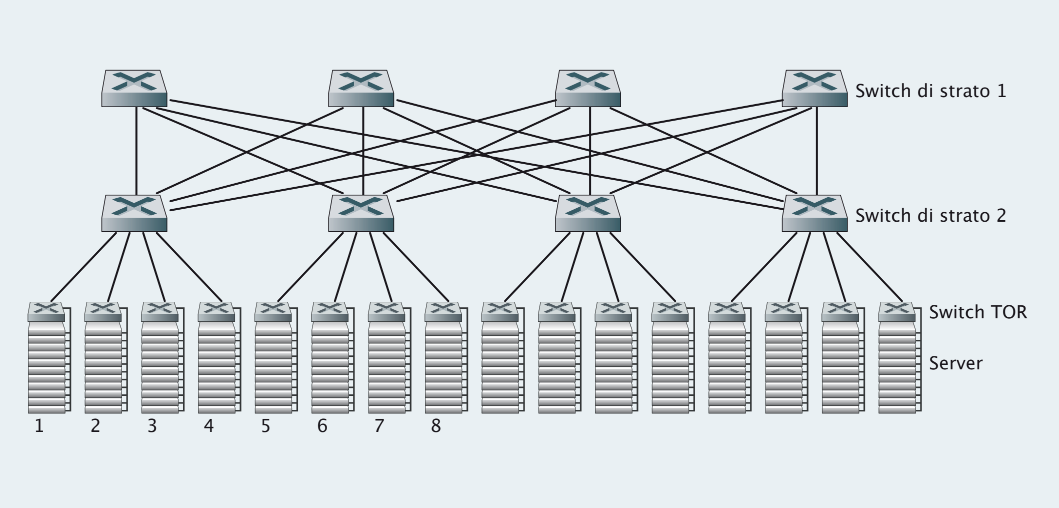


Figura **15** Architettura a connessione totale

Ogni switch della fila superiore si connette a tutti gli switch della fila intermedia : questo fa sì che il traffico interno non giunge mai ai router di accesso e che tra due switch della fila intermedia sono disponibili tanti percorsi quanti sono gli switch di prima fila. Tornando all'esempio dei 40 flussi, vediamo come in questo modo la larghezza di banda disponibile rimanga costante, dato che i due switch intermedi coinvolti hanno a disposizione 4 connessioni a 10 Gbps, ciascuna passante per uno dei 4 switch di prima fila.

Ovviamente, lo sfruttamento totale di queste particolari topologie comporta la creazione, da parte dei gestori di questi datacenter, di algoritmi di routing particolari e di protocolli specifici, che vengono tenuti gelosamente riservati all'interno delle organizzazioni.

5 Riassunto

Facciamo un riassunto di quanto abbiamo visto sino a ora e seguiamo tutte le attività che sono necessarie, all'interno della rete, per soddisfare la più semplice delle richieste: scaricare una pagina Web.

La figura 16 illustra la struttura di rete a cui ci riferiamo: uno studente collega il proprio portatile alla rete Ethernet della scuola, che è connessa a Internet tramite l'ISP Xfinity, e scarica una pagina dal sito di Google. Il server DHCP si trova, come normalmente accade, nel router della scuola, mentre il server DNS si trova presso Xfinity. Per semplicità, supponiamo che non vi sia NAT tra la LAN della scuola e l'ISP.

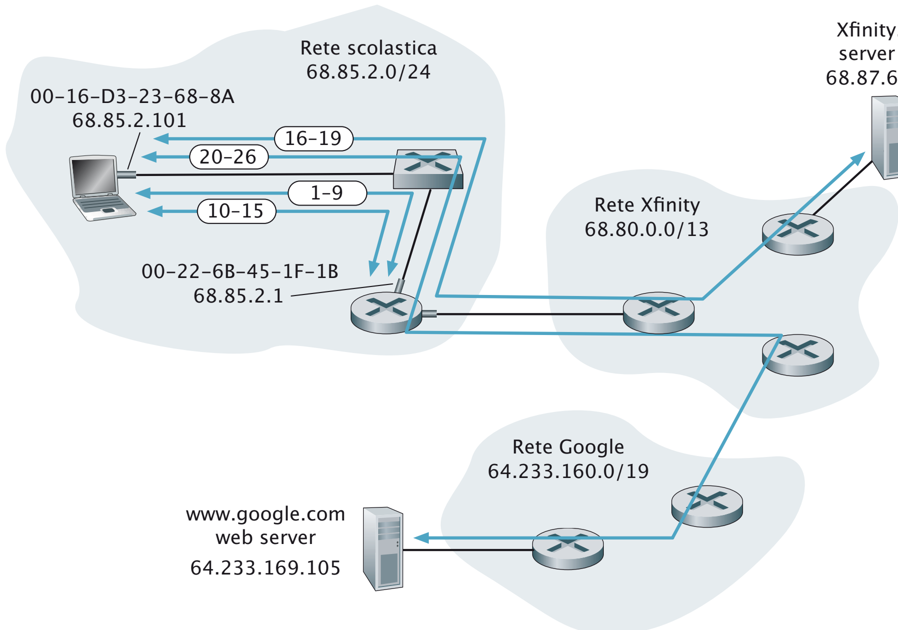


Figura **16**
Passaggi necessari per richiedere una pagina Web

DHCP, UDP, IP, Ethernet • Quando lo studente collega per la prima volta il portatile alla rete, non può fare nulla, perché non possiede un indirizzo IP valido per la LAN della scuola: la prima operazione necessaria, quindi, è ottenere un indirizzo IP e altre informazioni dal server DHCP locale. Ecco la sequenza delle operazioni:

1. il sistema operativo del portatile crea un messaggio *DHCP discovery*, lo inserisce in un segmento UDP con porta di destinazione 67 (server DHCP) e porta mittente 68 (DHCP client). Il segmento UDP è poi inserito in un datagram IP con indirizzo di destinazione broadcast 255.255.255.255. e indirizzo mittente 0.0.0.0, dato che, al momento, il portatile non possiede alcun indirizzo IP
2. il datagram viene a sua volta inserito in un frame Ethernet, che ha come MAC address di destinazione FF-FF-FF-FF-FF-FF, così da essere inviato a tutti i dispositivi connessi con lo switch, nella speranza che uno di essi sia il server DHCP. Il MAC address del mittente è quello dell'interfaccia di rete del computer, nel nostro caso 00-16-D3-23-68-8A
3. questo frame è il primo inviato dal portatile sulla rete. Lo switch lo inoltra su tutte le porte in uscita, inclusa quella collegata al router
4. il router riceve il frame contenente il *DHCP discovery* sull'interfaccia con MAC address 00-22-6B-45-1F-1B ed estrae il datagram IP dal frame Ethernet. L'indirizzo IP broadcast 255.255.255.255 indica che il datagram deve essere processato sul nodo - ricordiamo che il router contiene il DHCP server, ossia un'applicazione - e, quindi, il router provvede a demultiplare dal datagram il segmento UDP e a estrarre da questo il messaggio *DHCP discovery*

5. supponiamo che il server DHCP nel router possa allocare indirizzi contenuti nel blocco CIDR 68.85.2.0/24 e, per comodità, che questi siano contenuti nel blocco più grande di Xfinity 68.85.0.0/13. Supponiamo anche il server DHCP assegni al portatile l'indirizzo 68.85.2.101. Il router crea un messaggio *DHCP offer* contenente l'indirizzo IP proposto, l'indirizzo del server DNS (68.87.71.226), quello del router di default (68.85.2.1), ossia di se stesso, e la maschera di sottorete 255.255.255.0. Il messaggio DHCP è collocato all'interno di un segmento UDP, che a sua volta viene inserito in un datagram il quale, infine, viene incapsulato in un frame Ethernet. Quest'ultimo contiene il MAC address sorgente dell'interfaccia del router (00:22:6B:45:1F:1B) e quello di destinazione corrispondente all'interfaccia del portatile (00-16-D3-23-68-8A)
6. il frame Ethernet contenente il messaggio *DHCP offer* viene inviato in broadcast dal router allo switch, il quale lo dovrebbe riportare su tutte le proprie interfacce. Poiché, però, lo switch ha la funzione di autoapprendimento, si ricorda a quale delle sue interfacce è collegato l'host con MAC address 00-16-D3-23-68-8A, e quindi instrada il messaggio solo verso il portatile, contribuendo a mantenere più libera la rete
7. il sistema operativo del portatile spacchetta il frame, il datagram e il segmento, recupera il messaggio *DHCP offer* e risponde con il messaggio *DHCP request*, che impacchetta nella stessa sequenza, inviandolo in broadcast come i precedenti
8. il router riceve il frame e lo tratta come descritto al punto 4. Poi genera il messaggio *DHCP ack*, lo impacchetta e lo reinvia con le stesse modalità descritte al punto 6
9. Il portatile riceve e spacchetta il messaggio, e il client DHCP provvede a inserire nella propria configurazione gli indirizzi IP ricevuti, ossia il proprio indirizzo IP (65.85.2.101), quello del server DNS (68.87.71.226). Inserisce infine nella propria tabella di inoltro quello del suo router di default (68.85.2.1)

DNS, ARP • A questo punto, essendo connesso alla rete, lo studente può digitare nel browser l'indirizzo della pagina che vuole aprire (nel nostro caso *www.google.it*). Il browser inizia il processo di richiesta creando un socket TCP che serve a inviare la richiesta HTTP a Google.

Per poter creare il socket, il portatile deve conoscere l'indirizzo IP corrispondente a *www.google.it* e, quindi, deve utilizzare il protocollo DNS per ottenere il servizio di traduzione:

10. il sistema operativo del portatile crea un messaggio di query contenente la stringa *www.google.it*, e la inserisce in un segmento UDP con destinazione porta 53 (DNS server). Il segmento UDP viene inserito in un datagram IP con mittente 65.85.2.101 e destinatario 68.87.71.226
11. il datagram IP viene incapsulato in un frame Ethernet, che deve essere inviato, debitamente indirizzato a livello connessione, al router di default. Tuttavia, benché il portatile conosca l'indirizzo IP del router, non ne conosce il MAC Address: per ottenerlo, deve utilizzare il protocollo ARP
12. il portatile genera un messaggio di query ARP con destinatario 68.85.2.1, lo inserisce in un frame Ethernet con indirizzo broadcast (FF-FF-FF-FF-FF-FF) e lo invia allo switch, che, a sua volta, lo inoltra a tutti gli apparati connessi, compreso il router di default
13. il router riceve il frame contenente la query ARP sull'interfaccia della rete scolastica e trova che l'indirizzo 68.85.2.1 in esso contenuto corrisponde all'indirizzo IP di questa interfaccia: prepara, quindi, un messaggio di risposta ARP, indicando che il proprio MAC address corrispondente a questo indirizzo IP è 00-22-6B-45-1F-1B. Il messaggio viene inserito in un frame Ethernet con destinatario l'indirizzo 00-16-D3-23-68-8A, che corrisponde al portatile, e lo invia allo switch, che provvede a inoltrarlo
14. il portatile riceve dallo switch il frame contenente il messaggio di risposta ARP e ne estrae il MAC address del router di default
15. il portatile può, finalmente, indirizzare il frame Ethernet contenente la query DNS al MAC address del router di default. Notiamo che il datagram IP ha come destinatario l'indirizzo 68.87.71.226, ossia il server DNS, e il frame Ethernet che lo contiene ha destinatario il MAC address 00-22-6B-45-1F-1B, ossia il router di default. Il portatile invia il frame allo switch, che lo consegna al router di default

Routing intradominio e interdominio • A questo punto, per la prima volta dall'inizio, il portatile si connette con il mondo esterno alla propria sottorete:

16. il router di default riceve il frame e ne estrae il datagram contenente la query. Leggendo l'indirizzo IP del destinatario, determina dalla sua tavola di inoltro che il datagram deve essere inoltrato sulla connessione verso il router A della rete Xfinity: il datagram è incapsulato in un frame con indirizzo appropriato per questa connessione e spedito su di essa
17. il router A riceve il frame, estrae il datagram IP, esamina l'indirizzo di destinazione 68.87.71.226 e determina l'interfaccia di uscita a cui inoltrarlo verso il server DNS, utilizzando la propria tabella di inoltro, che è stata popolata dal protocollo intradominio della Xfinity (RIP, OSPF, IS-IS) e dal protocollo interdominio BGP
18. il datagram con la query DNS arriva al server DNS, che provvede a estrarre il messaggio. Poi cerca il nome *www.google.it* nel proprio database e trova il record relativo che contiene l'indirizzo IP corrispondente (64.233.169.105) supponendo che si trovi nella sua cache. In caso contrario, il server DNS dovrà a sua volta cercarlo presso i server DNS autorevoli. Il server DNS genera un messaggio di risposta contenente l'indirizzo IP, lo inserisce in un segmento UDP e il segmento in un datagram indirizzato al portatile (68.85.2.101). Questo datagram viene instradato in senso inverso rispetto alla query DNS, attraverso la rete Xfinity sino al router della scuola e da qui, attraverso lo switch, al portatile
19. il portatile estrae l'indirizzo IP per *www.google.it* e lo passa al browser

A5-04
Protocolli intra- e interdominio

TCP, HTTP • A questo punto, dopo molto lavoro, il laptop è pronto a connettersi al Web server *www.google.it*. Ecco gli ultimi passaggi:

20. finalmente in possesso dell'indirizzo IP del dominio, il sistema operativo può creare il socket TCP che sarà utilizzato per inviare il messaggio HTTP di richiesta della pagina. Quando il socket viene creato, il TCP del portatile deve come prima cosa effettuare l'handshake con il TCP di Google: come prima cosa, quindi, crea un segmento TCP SYN con destinazione porta 80 (Web server), lo inserisce in un datagram con indirizzo IP di destinazione 64.233.169.105 (*www.google.it*), mette il datagram in un frame con MAC address di destinazione 00-22-6B-45-1F-1B (router di default) e invia il frame allo switch
21. i router della scuola, di Xfinity e di Google inoltrano il datagram contenente il TCP SYN sino a www.google.it, utilizzando le proprie tabelle di inoltro, con le stesse modalità che abbiamo già visto nei passi da 16 a 18
22. il datagram arriva, infine, al server con indirizzo 64.233.169.105. Il TCP SYN ne viene estratto e demultiplato verso il socket di ingresso associato alla porta 80. Un socket di connessione viene creato per la connessione tra il server HTTP di Google e il portatile. Viene generato un segmento TCP SYNACK, inserito in un datagram IP indirizzato verso il portatile e, infine, viene incapsulato in un frame adatto al tipo di connessione presente tra il server HTTP e il router a esso più vicino
23. il datagram contenente il segmento TCP SYNACK è inoltrato attraverso le reti di Google, di Xfinity e della scuola, sino ad arrivare all'interfaccia Ethernet del portatile, che provvede a demultiplare il messaggio al socket TCP creato al punto 20, che accede allo stato di connessione
24. a questo punto il portatile è pronto a inviare byte al server di Google e il browser può inviare il messaggio di richiesta contenente l'URL da raggiungere: il messaggio è scritto nel socket, e il segmento TCP ripete il percorso descritto ai passi da 20 a 23
25. il server HTTP di Google legge dal socket il messaggio di richiesta http, crea un messaggio HTTP di risposta contenente nel corpo il codice HTML relativo alla pagina Web richiesta e manda il messaggio nel socket TCP
26. il messaggio di risposta ritorna al portatile. Il browser lo legge dal socket TCP, ne estrae il codice contenuto nel corpo e, finalmente, può mostrare sullo schermo la pagina Web

Concetti essenziali

- Qualunque apparato che gestisca un protocollo di livello 2 viene definito nodo.
- Il canale di comunicazione tra due nodi adiacenti viene definito connessione o link.
- I servizi di connessione fondamentali sono il framing, l'accesso alla connessione e la consegna affidabile.
- Esistono tecniche differenti per rilevare (e possibilmente correggere) gli errori: le più semplici sono però meno efficaci, soprattutto in caso di più errori compresenti nello stesso frame.
- Una delle tecniche di verifica degli errori più semplici è quella del controllo di parità.
- Le tecniche più complesse sono quella basata sul checksum e quella del controllo a ridondanza ciclica.
- La tecnica più usata è quella del controllo a ridondanza ciclica (CRC).
- La trasmissione a partire da un punto verso l'intera rete prende il nome di broadcast.
- Host e router, oltre agli indirizzi di livello rete, hanno anche indirizzi di livello connessione (MAC address) associati direttamente alle interfacce di rete.
- I MAC address non cambiano al cambiare della rete cui è connessa la NIC.
- Per indirizzare un datagram sono necessari sia l'indirizzo IP che il MAC address.
- Il protocollo che trasforma l'indirizzo IP nel MAC address prende in Internet il nome di ARP (Address Resolution Protocol).
- Il termine Ethernet indica una famiglia di tecnologie per reti locali, i cui standard ne definiscono le specifiche tecniche sia a livello fisico sia a livello connessione.
- Le varie versioni dell'Ethernet sono definite dallo standard IEEE 802.3.
- Nelle reti virtuali (VLAN), l'appartenenza di un utente a un gruppo è definita indipendentemente dalla sua posizione fisica nella rete.
- I datacenter sono strutture tecnologiche in cui sono concentrati apparati di rete e di calcolo in apposite infrastrutture e con collegamenti che ottimizzano le prestazioni del sistema.
- Le tecniche di bilanciamento (load balance) mirano a ottenere una efficiente ripartizione del carico di un sistema soggetto a grandi quantità di richieste provenienti dall'esterno
- I datacenter possono essere strutturati secondo un'architettura di tipo gerarchico o attraverso un'architettura a connessione totale

Test

1 Dire se le seguenti affermazioni sono vere o false.

Il nodo è definito come:

A un qualsiasi apparato hardware V F
B qualunque apparato di rete V F
C qualunque apparato che gestisca più protocolli contemporaneamente V F
D un qualsiasi apparato che gestisca un protocollo di livello 2 V F

2 Dire se le seguenti affermazioni sono vere o false.

La sigla NIC significa:

A Network Interface Card V F
B New Implementation Code V F
C Network Introductory Course V F
D National Internal Crew V F

3 Dire se le seguenti affermazioni sono vere o false.

Il servizio fondamentale del livello connessione si può sintetizzare come:

A la trasmissione di un messaggio via Web V F
B la garanzia della lunghezza costante dei frame V F
C lo spostamento di un datagram tra due nodi adiacenti V F
D l'incapsulamento del protocollo scelto a livello client V F

4 Dire se le seguenti affermazioni sono vere o false.

Il protocollo di accesso al mezzo trasmissivo prende il nome di:

A MAC V F

- **B** TMP V F
- **C** ACK V F
- **D** PTC V F

5 **Dire se le seguenti affermazioni sono vere o false.**
Sono possibili cause di errore di trasmissione in rete:
- **A** il contenuto del frame V F
- **B** le interferenze elettromagnetiche V F
- **C** il numero di porte degli switch V F
- **D** l'attenuazione del segnale V F

6 **Dire se le seguenti affermazioni sono vere o false.**
Il controllo di parità:
- **A** è una tecnica di rilevazione degli errori praticamente perfetta V F
- **B** è una tecnica di rilevazione degli errori con limitazioni V F
- **C** non permette mai di rilevare se uno 0 è stato trasformato in un 1 V F
- **D** è una delle tecniche di autocorrezione degli errori più usate V F

7 **Dire se le seguenti affermazioni sono vere o false.**
La tecnica di controllo degli errori di trasmissione basata sulla somma degli interi in cui è suddivisa la stringa da controllare prende il nome di:
- **A** Stringcount V F
- **B** Stringsum V F
- **C** Check-in V F
- **D** Checksum V F

8 **Dire se le seguenti affermazioni sono vere o false.**
Host e router, oltre agli indirizzi di livello rete, hanno anche:
- **A** indirizzi di livello spedizione V F
- **B** etichette di dispacciamento V F
- **C** tag di email V F
- **D** indirizzi di livello connessione V F

9 **Dire se le seguenti affermazioni sono vere o false.**
In una connessione:
- **A** il MAC address non cambia al cambiare della rete cui è connessa la NIC V F
- **B** il MAC address cambia al cambiare della rete cui è connessa la NIC V F
- **C** l'indirizzo IP non cambia ogni volta che l'host si connette a una rete diversa V F
- **D** l'indirizzo IP cambia ogni volta che l'host si connette a una rete diversa V F

10 **Dire se le seguenti affermazioni sono vere o false.**
Quando una NIC vuole mandare un frame a uno specifico destinatario:
- **A** incapsula l'indirizzo IP in una email V F
- **B** inserisce nel frame il MAC address relativo e lo invia sulla LAN V F
- **C** si collega in Wireless all'host e chiede conferma V F
- **D** invia una email al server e attende l'ACK relativo V F

11 **Dire se le seguenti affermazioni sono vere o false.**
La sigla ARP significa:
- **A** Artificial Random Process V F
- **B** Additional Relative Protocol V F
- **C** Add Risk Probability V F
- **D** Address Resolution Protocol V F

12 **Dire se le seguenti affermazioni sono vere o false.**
Una tabella ARP:
- **A** contiene sempre tutti i MAC address presenti nella sua sottorete V F
- **B** contiene gli indirizzi IP presenti nella sua sottorete V F
- **C** contiene i MAC address con cui l'host si è messo in comunicazione recentemente V F
- **D** serve per verificare se l'host sta inviando il MAC address corretto V F

13 **Dire se le seguenti affermazioni sono vere o false.**
I seguenti campi sono presenti in un frame Ethernet:
- **A** Preambolo, Header, Checksum V F
- **B** Indirizzo sorgente, Tipo, Dati V F
- **C** Tipo, Dati, CRC V F
- **D** Indirizzo destinazione, Server, Client V F

14 **Dire se le seguenti affermazioni sono vere o false.**
Le seguenti sono sigle valide di tipologie di Ethernet:
- **A** 10GBASE-X V F
- **B** 10BASE8-Y V F
- **C** 1000BASE-TX V F
- **D** 100BASE-RZ V F

15 **Dire se le seguenti affermazioni sono vere o false.**
All'interno dei datacenter, la ripartizione ottimale del carico avviene attraverso tecniche dette di:
- **A** Light Balance V F
- **B** Low Budget V F
- **C** Load Balance V F
- **D** Level Bottom V F

Sezione A

6 Reti wireless

Il capitolo tratta in modo approfondito un tema i cui sviluppi hanno letteralmente cambiato il modo di vivere, almeno nei Paesi più sviluppati: le reti wireless. Svincolando la postazione dell'utente da un punto preciso, quale la scrivania o il posto di lavoro, queste reti hanno permesso di creare un modo completamente nuovo di comunicare, rendendo l'informazione e l'accesso al Web indipendenti dalla posizione del fruitore. Con la rivoluzione del wireless, dalla mobilità della voce si è passati in pochi anni alla mobilità dei dati, includendo la multimedialità.
Vengono affrontati, nei paragrafi che seguono, gli aspetti delle diverse soluzioni disponibili e dello stato dell'arte, illustrando anche le criticità e le soluzioni grazie alle quali esse sono state, almeno assai sovente, superate.

1 Introduzione

Nel capitolo 1, parlando delle reti di accesso, abbiamo introdotto il concetto di connessione via radio, descrivendo per sommi capi le varie tipologie di reti wireless disponibili.

In questo capitolo possiamo entrare nel dettaglio e descriverne la struttura e il funzionamento, avendo già studiato l'architettura e i protocolli dei vari livelli di rete.

Benché molto diverse tra loro per scopi e struttura, le reti wireless possono essere descritte in modo univoco nelle loro componenti fondamentali: per comodità, nelle illustrazioni del presente capitolo utilizzeremo la grafica delle LAN, ma i principi sono validi anche per le grandi reti geografiche cellulari.

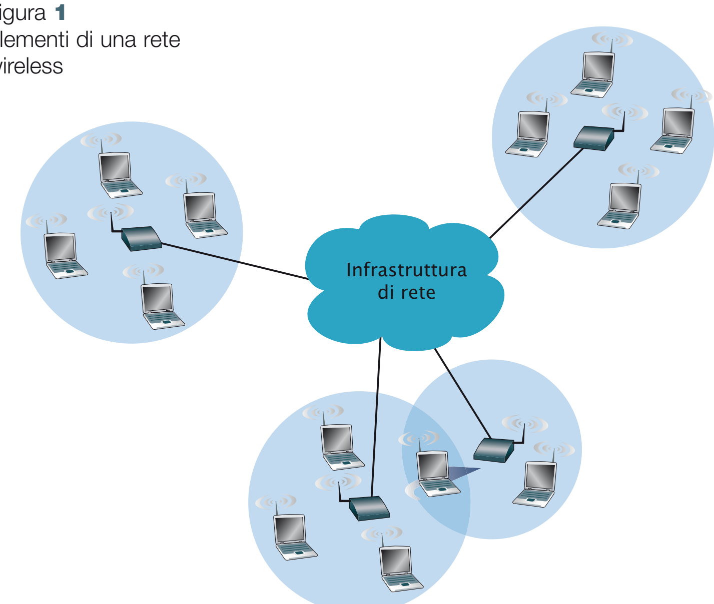


Figura **1**
Elementi di una rete wireless

Elementi base • In una rete wireless possiamo identificare alcuni elementi base:

- *Host wireless:* gli host sono gli apparati periferici su cui girano le applicazioni. Rispetto alle reti cablate, quelli delle reti wireless presentano una maggiore varietà: sono host wireless i PC di tutti i tipi, purché dotati di una NIC radio, gli smartphone, i tablet, i laptop, e a tutti i sistemi di connessione presenti, ormai, su qualunque nuovo modello di automobile
- *Connessioni wireless*: un host si connette a una ***stazione base*** (definita al punto successivo) o a un altro host tramite una *connessione wireless*. Esistono diversi tipi di connessione, che presentano differenti velocità di trasmissione e possono trasmettere su aree più o meno ampie: la figura 2 mostra queste caratteristiche riferite ai principali standard. Nella figura 1 le connessioni radio sono collocate nella parte periferica della rete, ma sono

utilizzate anche per l'interconnessione di router, switch e altri apparati che si trovano nel nucleo della rete. Lo studio di queste tecnologie di connessione, tuttavia, esula dai nostri obiettivi
- *Stazione base*: si tratta del componente chiave delle reti wireless. A differenza di host e connessioni, le stazioni base non hanno un componente corrispondente nelle reti cablate. Una stazione base ha il compito di scambiare dati con gli host a essa associati e di coordinare la trasmissione tra di essi. Si dice che un host è associato a una stazione base quando si trova all'interno del raggio di trasmissione della stazione e la utilizza come ripetitore per scambiare dati con reti di dimensioni più ampie. Le BTS delle reti cellulari e gli Access Point delle reti WiFi (di cui parleremo più avanti) sono esempi di stazioni base.

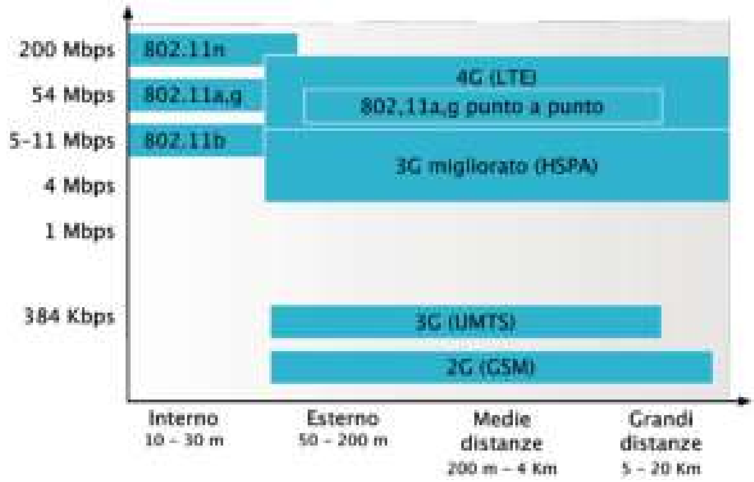


Figura **2**
Caratteristiche di connessione dei principali standard wireless.

Gli host associati a una stazione base operano ***in modo infrastruttura***, dato che tutti i servizi di rete tradizionali sono forniti dall'infrastruttura cui gli host si connettono tramite la stazione base. Nelle reti ***ad hoc***, invece, gli host non dispongono di un'infrastruttura e quindi devono provvedere direttamente a gestire i vari servizi, quali routing, assegnazione degli indirizzi e risoluzione del DNS.

Quando un host si sposta dall'area di copertura di una stazione base a quella di un'altra, cambia il punto di connessione verso l'infrastruttura: questo processo viene chiamato ***handoff*** (letteralmente *trapasso*) ed è causa di molte difficoltà nella gestione delle reti mobili. Lo spostamento dell'host da un punto all'altro della rete rende infatti necessari dei meccanismi di localizzazione istantanea della sua posizione nella rete e di gestione degli indirizzi a esso associati.

I vari elementi che abbiamo descritto possono combinarsi in molti modi, formando diverse architetture di reti wireless. Al livello più elevato, possiamo classificare le reti wireless secondo due criteri: l'esistenza di un'infrastruttura e il numero di tratte wireless che un pacchetto fa all'interno della rete. Ecco le tipologie esistenti:

- *Singola tratta, con infrastruttura*: queste reti hanno una stazione base connessa a una rete più ampia (ad esempio l'Internet); inoltre, la comunicazione tra l'host e la stazione base avviene con una singola tratta radio. Appartengono a questa classe la rete cellulare e le reti WiFi
- *Singola tratta, senza infrastruttura*: in queste reti non esiste una stazione base cablata che si connette a une rete wireless, ma uno dei nodi può coordinare le comunicazioni tra tutti i nodi che formano la rete. Le reti *802.11 ad hoc* e le *connessioni Bluetooth* appartengono a questa classe
- *Multitratta, con infrastruttura*: in questo caso, è presente una stazione base connessa a una rete più ampia, ma alcuni host wireless devono rimbalzare su altri host wireless per comunicare con la stazione base. Le ***reti mesh*** appartengono a questa tipologia
- *Multitratta, senza infrastruttura*: in queste reti non esiste stazione base, e i nodi possono rimbalzare i propri messaggi su altri nodi per raggiungere la propria destinazione. Si tratta di reti ancora allo stato sperimentale, soprattutto se i nodi sono mobili, data la complessità dei protocolli necessari per mantenere la connessione tra mittente e destinatario anche cambiando i nodi intermedi di rimbalzo

A6-01
Tipologie delle reti wireless

In questo capitolo, parleremo solo delle reti del primo tipo; alle altre tre classi è dedicato un approfondimento disponibile online.

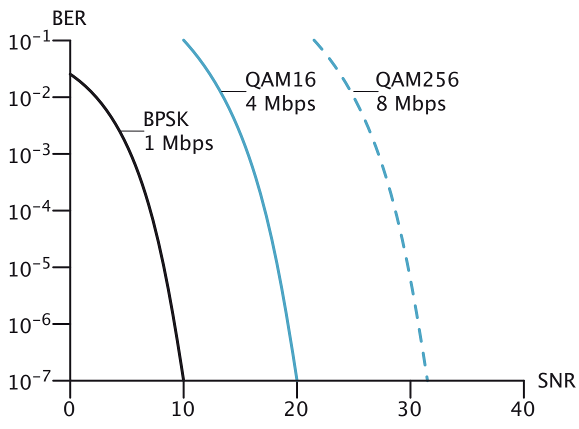
2 Caratteristiche delle connessioni wireless

Consideriamo una semplice rete cablata, formata da un certo numero di host collegati a uno switch: se la sostituiamo con una rete WiFi, le NIC Ethernet degli host lasceranno il posto ad altrettante NIC wireless, e lo switch verrà sostituito da un Access Point. Praticamente nessuna modifica sarà necessaria al livello rete e, quindi, ai livelli superiori. Le differenze si concentrano, dunque, al livello connessione oltre che, ovviamente, al livello fisico.

Possiamo trovare tre importanti differenze tra le connessioni nelle reti cablate e nelle reti wireless:

- *Riduzione del livello del segnale*: il campo elettromagnetico si attenua passando attraverso la materia e, anche nello spazio libero, la sua potenza decresce con la distanza, per cui il livello del segnale nella connessione diminuisce all'allontanarsi dei due nodi
- *Interferenza tra più sorgenti*: più trasmettitori nella stessa banda di frequenze interferiscono tra loro: ad esempio, i telefoni cordless DECT trasmettono sulla stessa banda a 2,4 GHz di alcune reti WiFi, per cui si può presumere che la loro compresenza in un'area peggiori le prestazioni di entrambi. Inoltre, il rumore elettromagnetico ambientale, causato da elettrodomestici, trasmissioni televisive, vicinanza di elettrodotti, ecc. causa ulteriori interferenze
- *Percorsi multipli*: noti comunemente con il termine ***multipath***, essi sono causati dalle riflessioni e rifrazioni che le onde elettromagnetiche subiscono incontrando ostacoli parzialmente riflettenti, quali pareti e oggetti metallici. La conseguenza è la generazione di più percorsi di diversa lunghezza tra sorgente e ricevitore, rendendo confuso il segnale

Figura **3**
Relazione tra tasso di errore e rapporto segnale/rumore.



Rapporto segnale/rumore • Da quanto detto, si capisce che ciò che un host riceve è la combinazione tra un segnale degradato rispetto a quello all'uscita del trasmettitore, dall'indebolimento dovuto alla distanza e dal multipath, e il rumore di fondo presente nell'ambiente.

Si dice rapporto segnale/rumore, abbreviato ***SNR*** (*Signal to Noise Ratio*), il rapporto tra la potenza del segnale ricevuto e il rumore di fondo. SNR è un numero puro espresso in decibel (dB).

È ovvio che, più alto è SNR, più facilmente il destinatario riuscirà a estrarre un'informazione priva di errori dal segnale ricevuto.

Se consideriamo il tasso di errore nella ricezione dei bit, abbreviato ***BER*** (*Bit Error Rate*), possiamo dire in prima approssimazione che in assoluto esso risulta tanto più basso quanto più alto è SNR, anche se il tasso di errore relativo dipende dalla tecnica di codifica e modulazione adottata nella trasmissione.

La figura 3 illustra il rapporto tra BER e SNR per tre modulazioni a diversa velocità.

Questa figura mette in evidenza alcune importanti caratteristiche delle connessioni wireless:

- Per qualunque schema di modulazione, maggiore è SNR, minore è BER. Più il mittente può accrescere il rapporto segnale/rumore, aumentando la potenza di trasmissione, più diminuisce la possibilità che un frame sia ricevuto in modo errato. Notiamo, comunque, che non ha senso accrescere SNR oltre un certo limite: diminuire il BER al di sotto di quello di una connessione cablata non porta alcun vantaggio pratico. Inoltre, l'aumento della

potenza di trasmissione causa una maggior richiesta di energia, con conseguenze negative per l'autonomia degli apparati mobili

- A parità di SNR, una modulazione con maggiore velocità di trasmissione presenta un BER in assoluto più elevato, ossia, a parità di tasso di errore, richiede SNR più alti. Dalla figura, vediamo che la modulazione BPSK, con velocità 1 Mbps, raggiunge un BER di 10-7 già con un SNR di 10 dB, mentre la QAM16, con velocità quadrupla, richiede circa 20 dB per ottenere lo stesso BER: con questo SNR il tasso di errore di BSPK è talmente basso da non essere praticamente misurabile. Da ciò si deduce che la scelta della modulazione dipende dalla velocità di trasmissione che si vuole mantenere e dal rapporto segnale/rumore disponibile, che a sua volta dipende dalle interferenze ambientali e dalla distanza fra mittente e destinatario
- È possibile utilizzare una *selezione dinamica* della tecnica di modulazione per adattare la velocità di trasmissione alle condizioni del canale. SNR, e quindi BER, possono cambiare in condizioni di mobilità dell'host o a causa di cambiamenti ambientali: modulazione e codifica adattiva sono utilizzate sia nelle reti WIFI, sia nei cellulari 3G per massimizzare la velocità di trasmissione in funzione di un BER massimo tollerabile

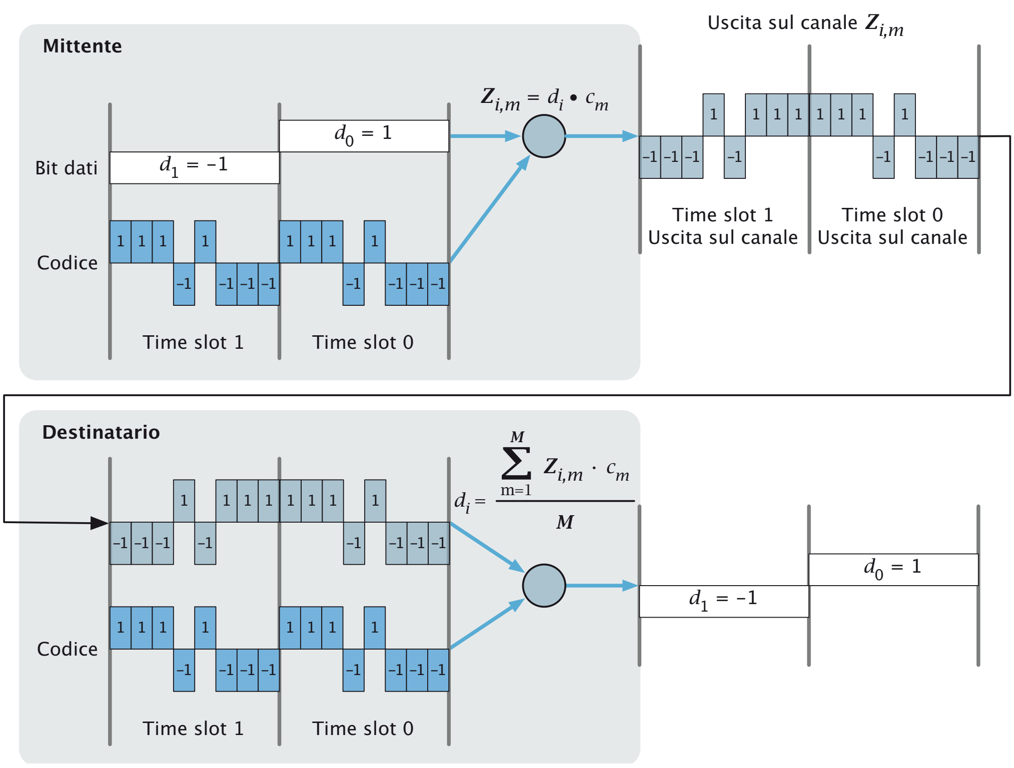
Un ultimo problema presente nelle reti wireless è il fenomeno del ***fading*** (letteralmente *affievolimento*), che provoca collisioni impossibili da rilevare da parte dei mittenti.
La figura 4 ci mostra una situazione di questo tipo: A e C inviano entrambi a B, ma la loro distanza reciproca è tale che ognuno non percepisce il segnale dell'altro e, quindi, né A né C si accorgono che il destinatario B, che riceve da entrambi un segnale sufficientemente forte, subisce l'interferenza tra le due diverse sorgenti di trasmissione.

Il protocollo CDMA • Nel capitolo precedente abbiamo visto come, quando gli host comunicano attraverso un mezzo condiviso, sia richiesto un protocollo in grado di impedire che mittenti multipli interferiscano tra loro. Abbiamo individuato tre classi di questi protocolli: ***a partizione di canale*** (*Channel Partitioning Protocols*), ***ad accesso casuale*** (*Random Access Protocols*) e ***a turno*** (*Taking Turns Protocols*).

Nelle reti cellulari e nelle LAN wireless, il protocollo normalmente utilizzato appartiene alla famiglia dei protocolli a partizione di canale ed è chiamato ***CDMA*** (*Code Division Multiple Access*).

In questo protocollo, ogni bit da inviare è codificato moltiplicandolo per un segnale (il *codice*) che cambia con una velocità molto maggiore di quella della sequenza originale di bit: questa velocità è detta ***chipping rate*** (letteralmente *tasso di frammentazione*).

Figura **4**
Schema di principio della codifica CDMA.



Facendo riferimento alla figura 4 e semplificando notevolmente quanto in effetti accade, presumiamo che il ritmo con cui i bit di dati raggiungono il codificatore CDMA definisca l'unità di tempo, ossia che ogni bit da trasmettere occupi esattamente uno slot temporale. Chiamiamo d_i il valore del bit dati nell'i-simo slot e indichiamo con -1 i bit a zero: in effetti, in un canale radio modulato, lo zero deve essere rappresentato da un valore negativo, in quanto la mancanza di segnale non può essere interpretata come un'informazione.

Ogni slot è suddiviso in *M* mini slot: nella figura 4, il valore di *M* è pari a 8, ma nella realtà viene utilizzata una frammentazione molto maggiore.

Ognuno di questi mini slot assume un valore -1 o 1, secondo la chiave del codice utilizzato: nel nostro caso, la sequenza del codice è 1,1,1,-1,1,-1,-1,-1. Il motore di codifica prende ogni mini slot e lo moltiplica per il valore del bit nello slot cui appartengono, generando un'uscita pari a $Z_{i,m} = d_i$. cm.

Come si vede in figura, se il bit da trasmettere è 1 (slot d_0), la sequenza di m valori dei mini slot rimane invariata, mentre se il bit è 0 (slot d_1) la sequenza assume valori complementari. Il destinatario prende le sequenze di mini slot e le confronta con il codice: se sono uguali, allora il bit contenuto nello slot temporale è 1, se sono complementari, allora il bit è 0.

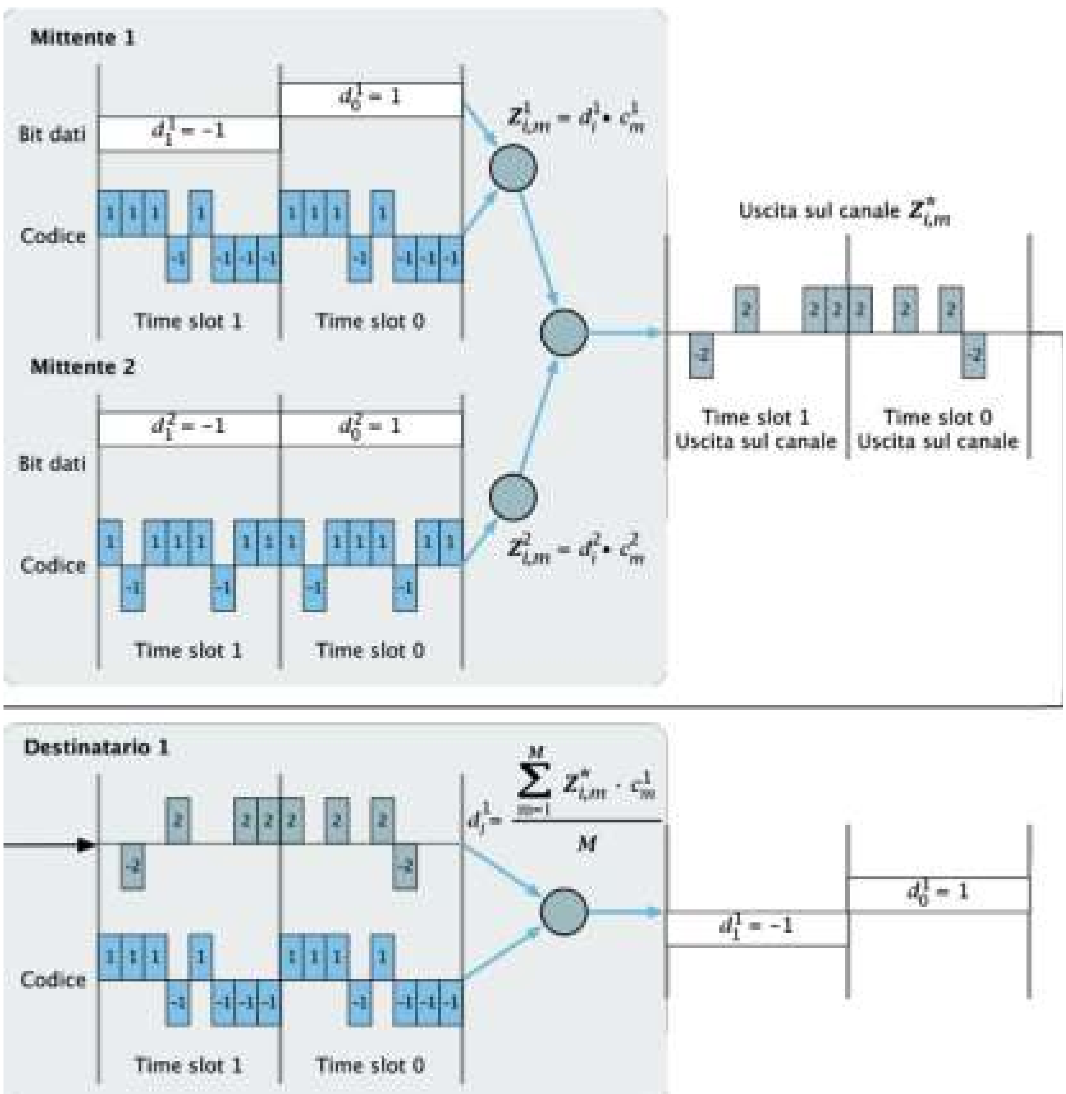


Figura **5**
Esempio di discriminazione tra due mittenti sovrapposti.

Ovviamente, questo processo aumenta notevolmente la quantità di bit inviati sul canale, dato che ogni bit di dati viene moltiplicato per il numero dei mini slot, come si vede chiaramente dalla formula dell'uscita.

Tuttavia, anche rallentando la velocità di trasmissione, il vantaggio della codifica con suddivisione dei bit da trasmettere è molto chiaro: più è grande la suddivisione, più è possibile perdere informazione senza che si generino errori nella decodifica.

Anche perdendo una grande quantità di bit, infatti, basta che una piccola parte dei mini slot ricevuti coincida con il codice o con il suo complemento per poter ricostruire correttamente il bit di informazione di partenza.

La figura 5 esemplifica questo meccanismo nel caso di un'interferenza additiva, come accade quando più host accedono allo stesso canale. Anche in questo caso vengono fatte notevoli semplificazioni, e si assume che le trasmissioni avvengano alla stessa velocità e con la stessa potenza. Notiamo che i valori 2 e -2 utilizzati nella figura rappresentano il maggior livello di ricezione dovuto alla somma di due segnali uguali.

3 802.11

Ormai le reti wireless hanno invaso il mondo e sono diventate la principale tecnologia di accesso all'Internet, grazie alla possibilità di connettere qualunque host senza necessità di accedere a un cablaggio.

Come sempre accade, agli inizi dello sviluppo sono stati generati diversi standard in concorrenza tra loro, ma quasi immediatamente è emerso un vincitore assoluto, che oggi è adottato universalmente: lo standard IEEE 802.11, conosciuto con il nome commerciale di WiFi.

Esistono diverse specifiche di questo standard, che si differenziano tra loro per la banda di frequenza utilizzata e per la velocità di trasmissione: in ordine di tempo sono apparse 802.11b, 802.11a, 802.11g, 802.11n e 802.11ac. La tabella 1 ne sintetizza le principali caratteristiche.

Standard	Gamma di frequenza	Velocità
802.11b	2,4 - 2,85 GHz	11 Mbps
802.11a	5,1 - 5,8 GHz	54 Mbps
802.11g	2,4 - 2,85 GHz	54 Mbps
802.11n	2,4 - 2,85 GHz e 5,1 - 5,8 GHz	300 Mbps
802.11ac	5,1 - 5,8 GHz	1,69 Gbps

Tabella **1** Caratteristiche dei principali standard IEEE 802.11.

Tutte le versioni condividono la maggior parte delle caratteristiche fondamentali:

- usano lo stesso protocollo di accesso
- hanno la stessa struttura dei frame a livello connessione
- hanno la capacità di ridurre la velocità di connessione per raggiungere maggiori distanze
- possono operare sia *in modo infrastruttura* sia *in modo ad hoc*

A livello fisico, invece, i vari standard presentano alcune fondamentali differenze (come si evince dalla tabella): variano sia le frequenze a cui operano, sia la velocità di trasmissione. Queste differenze sono state, sino a qualche anno fa, fondamentali per la scelta del tipo di rete da installare; oggi, con la diffusione degli standard ad alta velocità, il problema è molto meno sentito, anche in considerazione del fatto che una rete 802.11n o 802.11ac è in grado di connettere in modo trasparente anche host con specifiche inferiori.

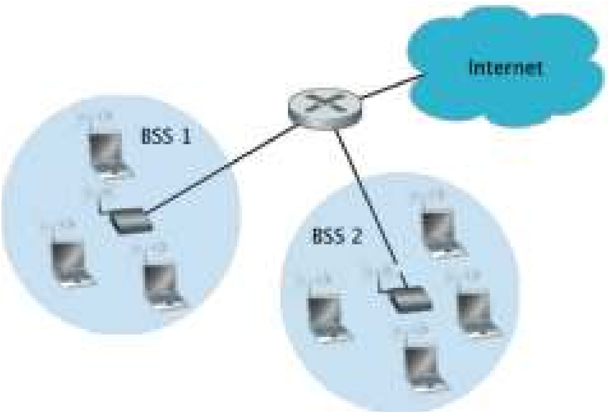


Figura **6**
Struttura di una rete 802.11.

Architettura • La figura 6 illustra i principali componenti di una rete 802.11 in modo infrastruttura. Il modulo fondamentale è chiamato ***BSS*** (*Basic Service Set*) e contiene uno o più apparati wireless e una stazione radio base o ***Access Point*** (***AP***).

Gli AP sono interconnessi tramite uno o più nodi, che provvedono anche alla connessione con l'Internet. In una tipica rete domestica, esiste un solo apparato che funge sia da AP che da router e permette anche la connessione di tratte Ethernet cablate.

Similmente all'Ethernet, ogni apparato wireless possiede un MAC address di lunghezza 6 byte, assegnato in maniera univoca dalla IEEE.

Nelle reti ad hoc, per la loro stessa natura, manca l'Access Point, e le stazioni si interconnettono direttamente. Questa configurazione non ha possibilità di connettersi al mondo esterno, ma permette di generare molto velocemente reti provvisorie fra dispositivi vicini anche in assenza di qualsiasi infrastruttura.

A causa della proliferazione di dispositivi mobili, le reti ad hoc rivestono attualmente grande interesse, ma dal nostro punto di vista possono essere considerate dei subset particolari del modo infrastruttura.

Canali e associazione • Per scambiare dati a livello rete, ogni stazione wireless deve essere associata a un AP. Quando l'AP viene installato, gli viene assegnato un identificatore, chiamato ***SSID*** (*Service Set Identifier*), che appare nella finestra di scelta della rete cui connettersi quando si associa la stazione.

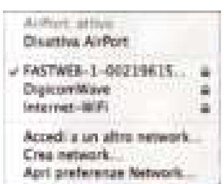


Figura **7**
Finestra di scelta della rete wireless sull'host.

È necessario, inoltre, assegnare all'AP un numero di canale, in modo da definire a quali frequenze esso opera. All'interno della banda di frequenza definita dallo standard, infatti, è disponibile un certo numero di canali con frequenza centrale equamente spaziata, ma parzialmente sovrapposti nella modulazione: se sono presenti più AP nella stessa area, è necessario configurarli in modo che i loro canali siano il più distanti possibile tra loro, in modo da evitare interferenze.

Per fare un esempio nella situazione più critica, ossia quella di una rete a 2,4 GHz (figura 8), troviamo che sono disponibili 11 canali all'interno della banda concessa di 85 MHz, e che solo a canali 1, 6 e 11 non si sovrappongono tra loro. Le specifiche più recenti, grazie al maggior numero di canali disponibili nella banda 5 GHz, alla possibilità delle AP di gestire più canali contemporaneamente e alla capacità di rilevare automaticamente la banda disponibile, rendono la situazione meno critica e più facilmente gestibile.

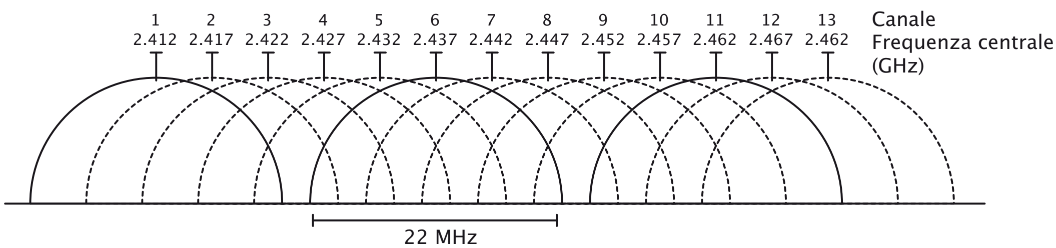


Figura **8**
Distribuzione dei canali 802.11 nella banda 2,4 GHz.

Malgrado ciò, la grande quantità di reti wireless installate genera situazioni - la cosiddetta "*giungla WiFi*" - che rendono necessari dei meccanismi di aggancio delle stazioni agli AP prescelti. Una tipica giungla WiFi è un condominio in cui ogni appartamento possiede la propria rete locale wireless, la cui portata supera ampiamente i confini dell'appartamento stesso.
Se un condomino vuole collegarsi alla propria rete e non a quella di qualche vicino, deve

associare i dispositivi che utilizza al proprio AP. Ciò significa che il dispositivo crea un filo virtuale tra sé stesso e l'AP e, quindi, entrambi sono in grado di riconoscersi all'interno della giungla.

Quando una stazione host si connette, ascolta tutte le AP disponibili e ne elenca l'SSID nella finestra di scelta (simile a quella di figura 7). L'utente sceglie a quale AP connettersi e la stazione si aggancia a esso, ignorando da quel momento tutte le altre possibilità, almeno sino a che l'AP è disponibile. Per ottenere ciò, lo standard 802.11 richiede che ogni AP trasmetta periodicamente dei frame di segnalazione, detti ***beacon frame*** (letteralmente *frame faro*), ognuno dei quali contiene l'SSID e il MAC address. La stazione effettua una scansione di tutti i canali, cercando i beacon frame di tutti gli AP, alcuni dei quali possono trasmettere sullo stesso canale.

802.11 non specifica un algoritmo di selezione da parte dell'host, lasciando la decisione ai singoli produttori. Normalmente, la scelta automatica avviene sulla base della potenza del segnale, ma esistono anche altre strategie che tengono conto del grado di occupazione delle varie AP. Se, ovviamente, l'SSID della rete cui l'host vuole collegarsi e noto, la scelta avviene manualmente. Questo processo di ascolto dei beacon frame è detto ***scansione passiva***.

Un host può anche effettuare una ***scansione attiva***, inviando un frame sonda che viene ricevuto da tutti gli AP all'interno della propria area di trasmissione: ricordiamo che, spesso, il raggio utile di trasmissione di una stazione è inferiore a quello di un Access Point, che dispone di maggior potenza e antenne più performanti. Quando un AP riceve il frame sonda, risponde con un frame contenete la propria identità.

Dopo aver selezionato l'AP cui associarsi, l'host invia un frame di richiesta di associazione, cui l'AP risponde con un frame di risposta di associazione: questo secondo handshake è necessario, in quanto l'AP non sa quale sarà la scelta dell'host tra i vari frame ricevuti.

A questo punto, l'host deve entrare nella sottorete cui appartiene l'Access Point scelto e, quindi, invia un messaggio *DHCP Discovery* che viene veicolato dall'AP al server DHCP, iniziando il processo descritto al capitolo 4.

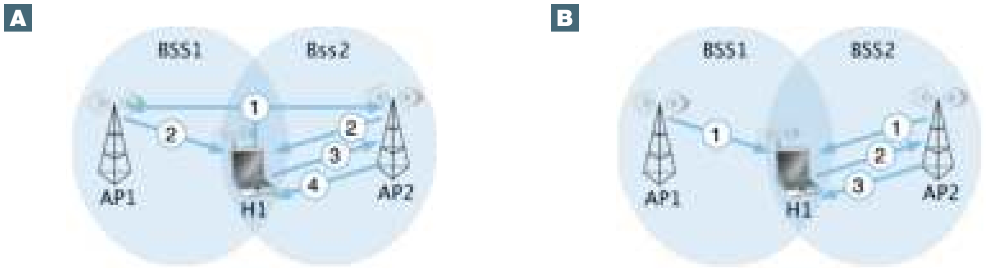


Figura **9**
(A) Scansione attiva e
(B) scansione passiva.

Autenticazione • Una volta effettuata connessione a livello fisico e a livello rete, per stabilire l'associazione con un particolare AP, alla stazione può essere richiesta un'autenticazione. Lo standard 802.11 fornisce diverse alternative per l'autenticazione e l'accesso. Le reti aziendali, spesso, utilizzano come elemento di autenticazione il MAC address dell'host, noto in quanto postazione fissa, mentre le reti aperte, quali quelle pubbliche, e le reti domestiche utilizzano generalmente nome utente e password.

In entrambi i casi, l'AP comunica con un server di autenticazione, utilizzando specifici protocolli.

Protocollo di accesso • Una volta che un host è associato a un Access Point, può iniziare a trasmettere e ricevere dati. Poiché, però, più host possono accedere contemporaneamente allo stesso AP sullo stesso canale, è necessario un protocollo di accesso multiplo (MAC, come abbiamo già visto nel capitolo 5) che coordini le trasmissioni. Ispirandosi al protocollo CSMA/CD utilizzato dell'Ethernet, lo standard 802.11 utilizza anch'esso un protocollo

ad accesso casuale, chiamato ***CSMA/CA***.

Come per il protocollo Ethernet, CSMA significa *accesso multiplo con rilevazione della portante*, mentre CA sta per ***Collision Avoidance***, ossia *elusione delle collisioni*.

Le differenze tra i due protocolli, malgrado il nome molto simile, sono significative: oltre a utilizzare tecniche di elusione anziché di rilevazione delle collisioni, 802.11 utilizza anche uno schema di riconoscimento/ritrasmissione a livello connessione, che l'Ethernet non prevede.

Struttura del frame • Benché il frame dello standard 802.11 condivida con l'Ethernet buona parte della propria struttura, esso contiene anche alcuni campi specifici all'uso in connessioni wireless.

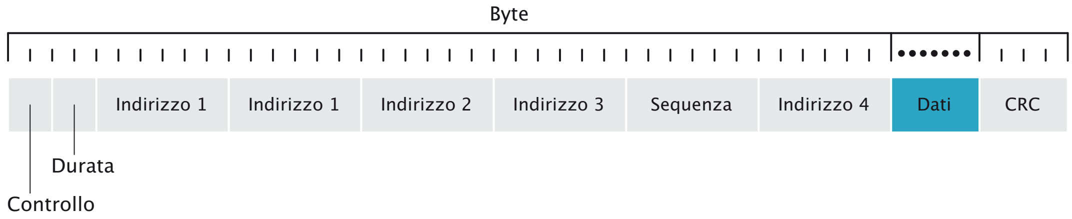


Figura **10**
Struttura del frame 802.11.

La descrizione è piuttosto articolata:

- *Controllo*: questo campo contiene molti sottocampi e verrà descritto a parte
- *Durata*: poiché il protocollo 802.11 consente alla stazione trasmittente di riservare il canale per un periodo di tempo sufficiente a inviare il frame e un ack, questo campo contiene il valore temporale della riserva
- *Indirizzo 1*: contiene il MAC address della stazione che deve ricevere il frame: se è inviato da un host, contiene l'indirizzo dell'Access Point di destinazione; se è inviato da un AP, contiene l'indirizzo dell'host
- *Indirizzo 2*: contiene il MAC address della stazione che invia il frame
- *Indirizzo 3*: poiché la BBS è parte di una sottorete e si connette alle altre sottoreti tramite un router o uno switch, questo campo contiene l'indirizzo dell'interfaccia del router che si connette a essa. Più avanti esamineremo meglio questa connessione
- *Sequenza*: quando una stazione riceve correttamente un frame da un'altra stazione, invia un ack di ritorno. Poiché gli ack possono andare perduti, spesso la stazione trasmittente invia più copie dello stesso frame. In questo caso, il campo contiene il numero progressivo del frame all'interno della sequenza di invii, in modo da permettere alla stazione ricevente di distinguere se si tratta di un frame nuovo o di una ripetizione
- *Indirizzo 4*: normalmente non utilizzato
- *Dati*: contiene il datagram IP o il pacchetto ARP. La capacità massima del campo è 2.312 byte ma, per coerenza con l'Ethernet, solitamente non ne contiene più di 1.500
- *CR*: come abbiamo visto nel paragrafo 2 del capitolo precedente, questo campo, lungo 4 byte, consente alla NIC del destinatario di effettuare il rilevamento e la correzione degli errori. Poiché le reti wireless soffrono di una maggior quantità di errori, questo campo risulta molto più utile che nel frame Ethernet

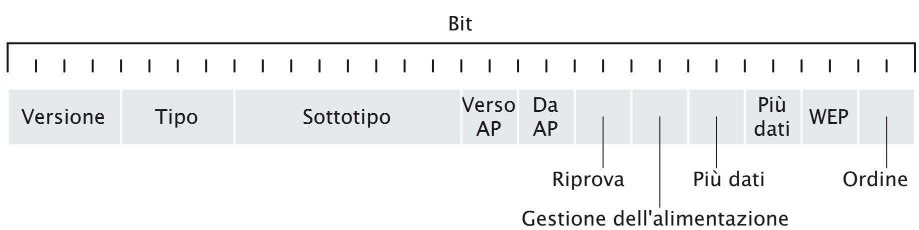


Figura **11**
Sottocampi contenuti nel campo Controllo.

Vediamo ora le suddivisioni del campo Controllo.

- *Versione*: indica la versione del protocollo utilizzata
- *Tipo*: identifica il tipo di frame (di controllo, di gestione, oppure di trasmissione dati, ecc.)
- *Sottotipo*: fornisce un'ulteriore discriminazione del tipo di frame: la somma dei due campi tipo e sottotipo identifica esattamente il frame
- *Verso AP* e *da AP*: definiscono i significati dei diversi campi indirizzo, che cambiano secondo chi trasmette il frame e secondo il tipo di rete (se è a infrastruttura o ad hoc):
- *Più frammenti*: indica se il datagram trasportato è diviso su più frame
- *Riprova*: indica quando si tratta di un reinvio e non di un nuovo frame
- *Gestione dell'alimentazione*: indica lo stato della gestione dell'alimentazione della stazione trasmittente alla fine dello scambio dei frame, come descritto nel capoverso "Gestione dell'energia" più oltre
- *Più dati*: serve a indicare alle stazioni in stato di risparmio energetico che vi sono frame bufferati
- *WEP*: indica se il frame è criptato
- *Ordine*: questo sottocampo è utilizzato solo quando è utilizzata la trasmissione dei frame in stretto ordine sequenziale

Internetworking • Per capire meglio il significato del campo *indirizzo 3*, facciamo un esempio di internetworking, con l'aiuto della figura 12.

Ognuno dei due AP mostrati nella figura gestisce un certo numero di host ed è direttamente collegato al router, che provvedere all'instradamento verso il mondo esterno e tra le due BSS.

Consideriamo la trasmissione di un datagram dal router all'host H1: gli Access Point, come gli switch, sono apparati che funzionano a livello rete, quindi non gestiscono gli indirizzi IP e risultano totalmente trasparenti al router, che vede solo H1 come un host appartenente alla specifica sottorete.

Il router, che conosce l'indirizzo IP di H1, utilizza ARP per determinarne il MAC address, come in una normale rete Ethernet cablata.

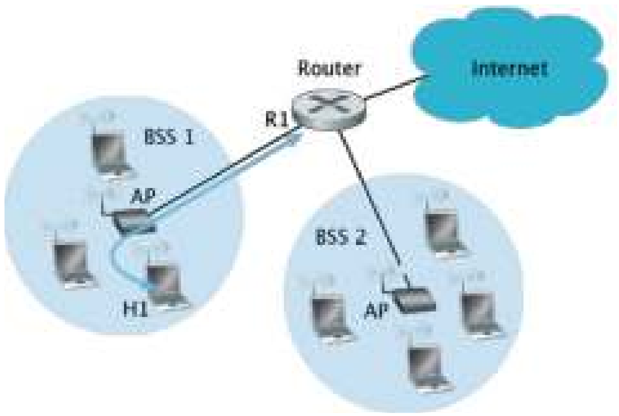


Figura **12**
Un Access Point risulta trasparente al livello rete.

Dopo averlo ottenuto, la NIC R1 del router incapsula il datagram in un frame Ethernet, in cui il campo *indirizzo sorgente* contiene il MAC address della NIC R1 e il campo indirizzo di destinazione contiene il MAC address dell'host H1.

Quando il frame Ethernet raggiunge l'Access Point, questo lo converte da 802.3 a 802.11, prima di trasmetterlo sul canale wireless: nel campo *indirizzo 1* inserisce il MAC address dell'host, mentre nel campo *indirizzo 2* inserisce il proprio MAC address (altrimenti l'host non sarebbe in grado di connettersi). L'indirizzo del router, comunque necessario, viene quindi inserito nel campo *indirizzo 3*.

Quando l'host deve rispondere al frame ricevuto, ne genera uno che contiene nel campo *indirizzo 1* il MAC address dell'AP, nel campo *indirizzo 2* il proprio e nel campo *indirizzo 3* mantiene il MAC address del router, così come gli è stato inviato. L'Access Point che riceve il frame 802.11 lo converte in un frame 802.3, in cui il campo *indirizzo sorgente* è mutuato da *indirizzo 2* e il campo indirizzo di destinazione è mutuato da *indirizzo 3*. In questo modo l'AP rimane completamente trasparente al router.

Mobilità della sottorete • Spesso il raggio di azione di un Access Point non è sufficiente per le necessità di una struttura di una certa dimensione. Per questa ragione è necessario utilizzare più AP - e quindi generare più BSS - all'interno di una stessa sottorete IP, il che comporta alcuni problemi di mobilità tra una BSS e l'altra.

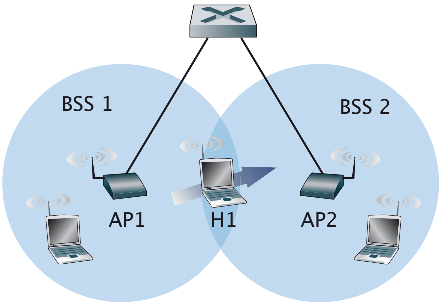


Figura **13**
Presenza di più BSS all'interno di una stessa sottorete.

La figura 13 mostra la situazione appena descritta: come si vede, le due BSS non sono collegate tra loro mediante un router, come avviene invece nella figura 6, ma attraverso uno switch che, non gestendo gli indirizzi IP, le accomuna entrambe nella stessa sottorete. Se l'host H1 passa da una BSS all'altra, può portare con sé sia l'indirizzo IP che tutte le connessioni TCP in corso, senza interruzione del servizio, mentre se si trattasse di due distinte sottoreti, sarebbe necessario un cambio di indirizzo IP, il che farebbe crollare la connessione.

Il passaggio da una BSS all'altra avviene molto semplicemente: quando H1 si allontana da AP1, rileva l'indebolimento del segnale e inizia la scansione per rilevare un segnale più forte. Appena riceve i beacon frame da AP2, che è stato configurato con lo stesso SSID di AP1, si sgancia da AP1 e si associa ad AP2, conservando il proprio indirizzo IP, che viene gestito dal livello superiore, e mantenendo le connessioni TCP attive.

Resta lo switch, che non sa dello spostamento di H1 da una porta all'altra, ma che, grazie alle proprie capacità di autoapprendimento (che abbiamo visto nel capitolo 5) è in grado di aggiornare le proprie tabelle di instradamento e, quindi gestire automaticamente il passaggio da una BSS all'altra. Questo metodo, tuttavia, è adatto solo per spostamenti sporadici, e causa comunque indirizzamenti errati da parte dello switch durante lo spostamento. La soluzione, non elegante ma funzionale, consiste nel far sì che AP2, all'associazione di H1, invii un frame Ethernet broadcast contenente l'indirizzo sorgente di H1, in modo che lo switch possa aggiornare immediatamente la tabella di instradamento, facendo sì che H1 possa essere raggiunto tramite AP2.

Caratteristiche avanzate • Le reti wireless possiedono anche delle caratteristiche avanzate che non sono completamente definite nello standard 802.11, ma che sono rese possibili dai meccanismi in esso specificati.

Abbiamo visto, al paragrafo 1, che esistono differenti tecniche di modulazione, adatte a diverse situazioni di rumore: se un host si trova vicino all'AP e ha un elevato SNR, può usare una modulazione che gli consente alta velocità di trasferimento, pur mantenendo un basso tasso di errore. Ma, se l'host si allontana dall'AP, il SNR decresce con la distanza e, se viene mantenuta la stessa tecnica di modulazione, il tasso di errore aumenta sino a che nessun frame è più ricevuto correttamente.

Per questa ragione, alcune implementazioni commerciali hanno la capacità di adattare il tipo di modulazione al livello fisico, basandosi sulle caratteristiche istantanee del canale. Se una stazione invia due frame successivi senza ricevere un ack, ossia un'indicazione implicita di un alto tasso di errore, la trasmissione decade al livello immediatamente inferiore; se dieci frame di fila ricevono un ack, la velocità di trasmissione risale al livello immediatamente superiore.

Questo approccio è simile al controllo di congestione del protocollo TCP: quando le condizioni sono buone, la velocità di trasmissione è progressivamente aumentata, sino a che non accade qualcosa di negativo, che inverte la tendenza e causa una diminuzione della velocità.

Gestione dell'energia • L'autonomia è molto importante per un dispositivo mobile. Per questo lo standard 802.11 è dotato di una gestione dell'alimentazione che consente alle stazioni di minimizzare il tempo di attivazione delle funzioni di scansione, trasmissione e ricezione e la necessità di avere attivate altre funzioni del dispositivo. Un host è in grado di alternare gli stati di veglia e di sonno, indicando esplicitamente all'AP che sta per entrare in standby mettendo a 1 il sottocampo *gestione dell'alimentazione*. Un timer interno fa in modo che si risvegli in tempo per rispondere al beacon frame successivo. Nel frattempo, l'AP sa che non deve inviare ulteriori frame all'host, e provvede a inserirli in un buffer, pronti per una trasmissione successiva.

L'host si sveglia immediatamente prima dell'invio del beacon frame da parte dell'AP. I

beacon frame contengono una lista degli host che hanno frame in attesa nel buffer: se non fa parte della lista, l'host torna immediatamente in sonno, altrimenti richiede all'AP l'invio dei frame nel buffer inviando un messaggio di interrogazione.

Considerando che l'intervallo tra due beacon frame è di 100 ms e che risvegli, ricezione e interrogazione durano circa 250µs ciascuno, si vede come un dispositivo, se non deve ricevere o inviare messaggi, resti in standby per il 99% del tempo, minimizzando così il consumo di energia.

4 Accesso da rete cellulare

Sinora abbiamo parlato di reti wireless in cui la mobilità è intesa come possibilità di connettere all'Internet un host trasportabile in presenza di un Access Point WiFi, ma senza permettere all'host di muoversi con continuità tra Access Point diversi e nel contempo mantenere la connessione.

Per ottenere questo risultato, si è scelta la strategia di estendere le capacità delle reti cellulari, in modo da renderle capaci di supportare, oltre alla telefonia, anche l'accesso all'Internet a una velocità ragionevole e garantendo la continuità delle sessioni TCP anche agli utenti che si muovono su mezzi di trasporto veloci.

Panoramica sulle reti cellulari • Tra i diversi tipi di reti cellulari, esiste uno standard che copre oltre l'80% degli utenti in tutto il mondo: esso è chiamato GSM ed è stato sviluppato in Europa negli anni '80 con l'obiettivo di unificare le reti nazionali, allora incompatibili tra loro e di basse prestazioni. Il GSM ha avuto un grande successo negli anni '90, espandendosi fuori dall'Europa e guadagnando sempre in funzionalità. Per comodità, si indicano come 1G (prima generazione) le reti analogiche, come 2G le reti GSM digitali con il solo supporto della voce, come 2,5G le reti GSM con anche il supporto dati, come 3G le reti con capacità estesa di gestione dati, con velocità di connessione sino a 3 Mbps e, infine, come 4G le reti di ultima generazione, in grado di gestire file multimediali con velocità sino a 100 Mbps in movimento e 1 Gbps in posizione statica.

Architettura delle reti 2G • Il termine cellulare si riferisce al fatto che la regione di copertura della rete è suddivisa in un certo numero di aree, dette celle, ognuna delle quali dipende da una stazione base, detta ***BTS*** (Base Transceiver Station), la quale comunica in

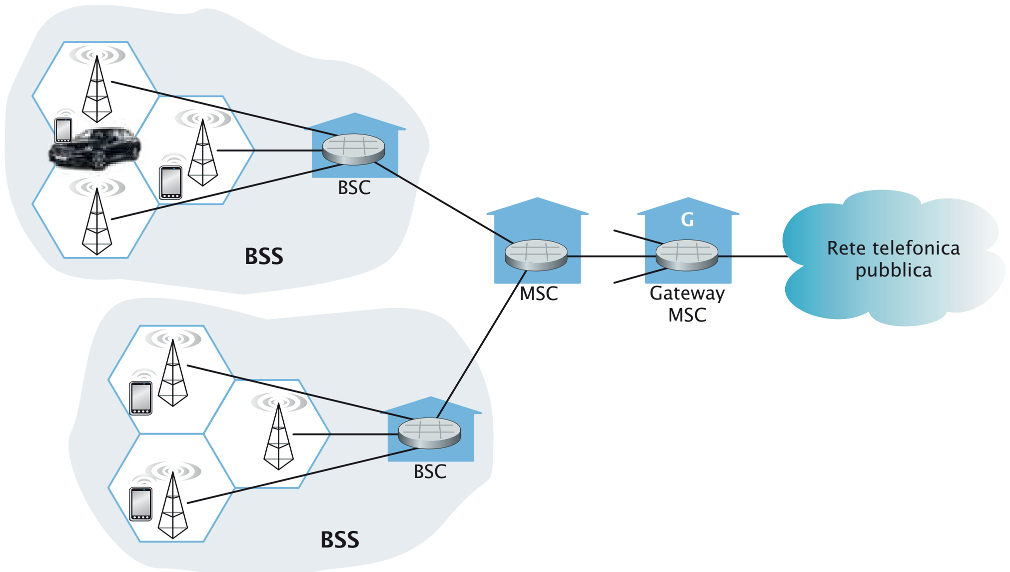


Figura **14**
Struttura di una rete cellulare 2G.

trasmissione e ricezione con le stazioni mobili che si trovano all'interno della cella. Le dimensioni delle celle dipendono dalla potenza delle BTS, dalla loro posizione rispetto a eventuali ostacoli e dalla potenza dei dispositivi mobili (figura 14).

Lo standard GSM per le reti voce utilizza una combinazione di FDM e TDM per l'interfaccia di comunicazione: riprendendo le definizioni date nel capitolo 1, diciamo che nei sistemi combinati FDM/TDM, il canale è suddiviso in un certo numero di sottobande di frequenza e, all'interno di ogni sottobanda, il tempo è suddiviso in frame e slot.

Un ***BSC*** (*Base Station Controller*) è in grado di gestire diverse decine di BTS e provvede ad alcune funzioni fondamentali:

- alloca i canali radio delle BTS verso i dispositivi mobili
- gestisce il ***paging***, ossia la scelta della cella in cui ogni utente si trova in un dato momento
- gestisce l'***handoff***, ossia il passaggio di un dispositivo dalla copertura di una cella a quella di un'altra

L'insieme di un BSC e dei BTS ad esso connessi forma un ***BSS*** (*Base Station System*).

I BSC sono connessi con un nodo centrale detto ***MSC*** (*Mobile Switching Center*), che provvede ad autorizzare le connessioni dei dispositivi mobili, alla contabilizzazione e a gestire gli handoff. Generalmente, un MSC riesce a gestire circa 200.000 utenti: quindi ogni carrier deve disporre di un certo numero di MSC, alcuni dei quali fungono da gateway di connessione tra la rete cellulare e la rete telefonica fissa.

Architettura delle reti 3G • Un dispositivo mobile che si connette all'Internet attraverso la rete cellulare deve, ovviamente, poter utilizzare lo stack TCP/IP nella sua interezza, compreso il livello fisico, ma attraverso la connessione dati delle reti cellulari. Ciò non è semplice, dato che queste ultime hanno standard in continua evoluzione e sono piuttosto diversificate in relazione alla regione del mondo in cui ci si trova. Per semplicità, nel seguito descriveremo lo standard ***UMTS*** (*Universal Mobile Telecommunications Service*), che è quello adottato in tutta Europa.

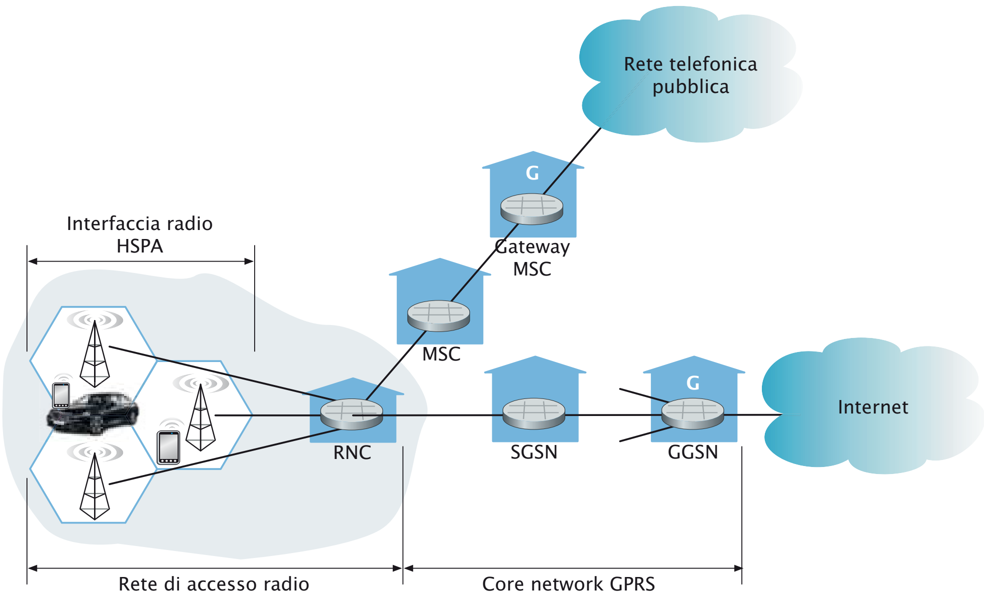


Figura **15**
Struttura di una rete cellulare 3G.

Osservando la figura 15, notiamo come l'architettura delle reti 2G sia stata mantenuta integralmente, e siano stati aggiunti ulteriori nodi per la gestione dei dati: a causa dell'estensione della rete cellulare esistente, infatti, l'obiettivo primario è stato quello di lasciare intatta la rete

voce, aggiungendo in parallelo le funzionalità dati. Questo ha comportato alcune complessità strutturali che si sarebbero potute altrimenti evitare, ma ha garantito la totale compatibilità con l'architettura di distribuzione del segnale già esistente.

Una significativa differenza tra le reti 2G e le UMTS sta nel fatto che queste ultime, anziché usare lo schema misto FDM/TDM, utilizzano una tecnica CDMA (di cui abbiamo parlato nel paragrafo 2), cosa che ha richiesto l'installazione di una rete di accesso parallela alla rete voce e l'affiancamento dei BSC con gli ***RNC*** (*Radio Network Controller*), che hanno le stesse funzioni voce, cui aggiungono le funzioni dati.

Anche il nucleo della rete 3G prevede dei nodi centrali, chiamati ***SGSN*** (*Serving GPRS Support Node*) e i relativi gateway, chiamati GGSN verso l'Internet. La funzione degli SGSN è quella di mantenere le informazioni di localizzazione dei nodi mobili attivi e di effettuare l'instradamento dei datagram tra host mobili e verso i GGSN.

Questi, a propria volta, provvedono a mantenere collegati, attraverso l'Internet, i diversi SGSN dello stesso carrier.

Gli SGSN sono connessi anche con i corrispondenti MSC, per gestire l'autorizzazione e l'handoff degli utenti dati.

Reti 4G • Si tratta di reti con un'architettura molto diversa dalle generazioni precedenti, frutto dell'implementazione degli standard ***LTE*** (*Long Term Evolution*), che presentano significative innovazioni.

Innanzi tutto, rispetto alla reti 3G, scompare la separazione tra il circuito voce (a commutazione di circuito) e il circuito dati (a commutazione di pacchetto). Essi vengono sostituiti da una rete totalmente IP, in cui sia voce che dati sono trasportati da datagram IP.

Per ottenere ciò, è stata sviluppata un'architettura chiamata ***EPC*** (*Evolved Packet Core*).

Poiché il modello di servizio di tipo best effort fornito dal protocollo IP non è intrinsecamente adatto alle stringenti richieste del VoIP, a meno che le risorse della rete siano attentamente gestite in modo da evitare le congestioni, anziché reagire ad esse, il primo scopo di EPC è quello di utilizzare le reti in modo da ottenere la massima qualità di servizio.

EPC, inoltre, opera una netta distinzione tra il piano del controllo di rete e quello dei dati utente.

Benché presenti caratteristiche avanzate, questo standard può utilizzare diversi tipi di reti di accesso: tra queste, le reti 2G e 3G, che in questo modo non vengono rese obsolete e possono essere collegate al nucleo di rete.

LTE • Le reti radio di accesso LTE usano una combinazione FDM/TDM, detta ***OFDM*** (*Orthogonal Frequency Division Multiplexing*). Il termine *ortogonale* deriva dal fatto che i segnali inviati sui diversi canali sono strutturati in modo da interferire molto poco tra loro, anche quando la spaziatura dei canali è molto stretta.

Nella LTE, ogni nodo mobile attivo ha allocato uno o più time slot da 500 μs in uno o più canali: più slot ha allocati, sia sullo stesso che su diversi canali, più alte sono le prestazioni raggiunte dal nodo. La riallocazione degli slot tra i nodi collegati alla rete può essere effettuata anche ogni ms. Similmente a quanto descritto nel paragrafo 3 riguardo alle reti WiFi, LTE può utilizzare anche diversi schemi di modulazione per cambiare la velocità di trasmissione. La massima velocità che può raggiungere un dispositivo LTE è 100 Mbps in trasmissione e 50 Mbps in ricezione, molto più alta di qualsiasi connessione ADSL cablata.

Nelle reti LTE, la decisione sull'assegnazione dei time slot a un particolare utente è flessibile, ed è determinata da particolari algoritmi di ***scheduling*** (letteralmente programmazione), che combinano il protocollo di livello fisico con le condizioni del canale fra trasmittente e ricevente e scelgono a quali apparati inviare i pacchetti, basandosi sulle condizioni di canale: in questo modo l'RNC può sfruttare al massimo le possibilità istantanee del mezzo trasmissivo.

Infine, la definizione di diverse priorità tra gli utenti e la disponibilità di diversi livelli di servizio possono essere utilizzati per gestire diverse velocità di connessione tra i singoli utenti.

Figura **16**
Distribuzione delle reti cellulari nel mondo.
Le aree in rosso dispongono di LTE.

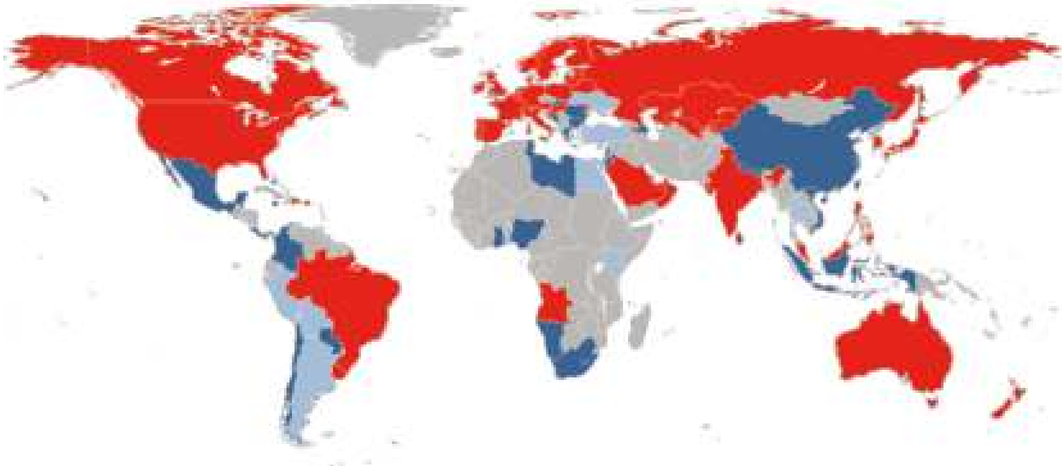


Figura **17**
Il logotipo Bluetooth.

Bluetooth

Nel primo biennio abbiamo visto come le reti possano dividersi, secondo le loro dimensioni e la loro portata, in LAN, MAN e WAN. Esistono, però, anche reti wireless di dimensioni ancora più ridotte delle LAN: le ***PAN*** (*Personal Area Network*) il cui scopo è rimpiazzare i cavi con cui si connettono tra loro computer, periferiche, smartphone e tablet.

Lo standard ormai universalmente adottato per le PAN è ***IEEE 802.15.1***, comunemente detto ***Bluetooth***. Questo nome deriva dall'anglicizzazione dall'epiteto con cui è indicato il re vichingo Harald, vissuto nel X secolo, che ha unito in un unico regno i popoli scandinavi: nelle intenzioni degli svedesi della Ericsson, sviluppatori della tecnologia, il Bluetooth avrebbe dovuto, allo stesso modo, unificare il sistema di connessione intorno al PC, cosa realmente accaduta.

Bluetooth opera nella banda di frequenze attorno ai 2,4 GHz, con una particolare modulazione TDM chiamata ***FHSS*** (*Frequency Hopping Spread Spectrum*, letteralmente *spettro diffuso a salti di frequenza*). Durante ogni slot temporale, che dura 625 μs, il mittente trasmette su uno dei 79 canali disponibili, ma il canale cambia da uno slot all'altro secondo una sequenza nota, anche se apparentemente casuale. La velocità di comunicazione può arrivare sino a 4 Mbps.

Poiché nella specifica 802.15.1 manca l'equivalente di un Access Point, si tratta di una rete di tipo ad hoc, priva di infrastruttura, in cui gli apparati devono organizzarsi tra loro. Il primo passo è la generazione di una *picorete* formata da un massimo di otto apparati, uno dei quali viene designato come *master*, mentre gli altri operano come *slave*. Il master regola effettivamente la rete:

- il suo clock determina quello dell'intera rete
- può trasmettere su uno qualunque dei canali dispari
- uno slave può trasmettere solo dopo che il master gli ha inviato una comunicazione, e può rivolgersi solo al master stesso.

Oltre al master ed agli slave, nella rete possono esistere anche un massimo di 255 apparati, detti ***parcheggiati***, che non possono comunicare sino a che il loro status non viene cambiato dal master. Ovviamente, se nella picorete sono già attivi sette slave, e ne viene fatto entrare uno nuovo, uno dei precedenti deve assumere lo status di parcheggiato, lasciando posto al nuovo slave.

Concetti essenziali

- La tecnologia wireless permette di comunicare senza dover connettere gli apparati a un sistema di cablaggio fisico.
- Gli elementi base per una rete wireless sono l'host wireless, le connessioni wireless e le stazioni base.
- Le reti wireless possono essere classificate in funzione della loro infrastruttura e del numero di tratte che un pacchetto percorre nella rete.
- Le trasmissioni sono rese critiche dal peggioramento del rapporto tra segnale e disturbo oltre che dall'affievolimento della potenza del segnale al crescere della distanza tra sorgente e ricevitore.
- Lo standard di riferimento per la tecnologia wireless è lo IEEE 802.11, noto commercialmente come WiFi.
- Un modulo fondamentale per le comunicazioni wireless è formato da un Access Point e da uno o più apparati wireless.
- Esistono reti in cui manca l'Access Point e gli apparati si connettono direttamente.
- Per scambiare dati a livello rete ogni stazione wireless deve fare riferimento a un Access Point.
- Il protocollo di accesso per le reti wireless prende il nome di CSMA/CA.
- I frame usati nelle comunicazioni wireless assomigliano ai frame Ethernet ma hanno anche alcuni campi specifici.
- Il raggio di azione di azione di un Access Point è comunque sempre limitato.
- La comunicazione in cui gli host si trovano in movimento rispetto alle BSS è problematica e richiede particolari strategie.
- Gli apparati mobili possono trovarsi a lungo lontani da sorgenti di energia: per questo motivo, la loro progettazione è mirata alla ricerca di soluzioni a minimo dispendio energetico, per aumentarne l'autonomia di funzionamento.
- Le reti wireless sono state spesso realizzate come estensioni delle reti di telefonia cellulare.
- Le architetture 2G, 3G, 4G e LTE delle reti cellulari sono caratterizzate da complessità e funzionalità crescenti e dal fatto che ogni livello superiore progettato in modo da conservare la compatibilità di funzionamento verso i livelli inferiori.

Test

1 Dire se le seguenti affermazioni sono vere o false.
I seguenti sono elementi componenti una rete wireless:

A stazioni base V F
B guest wireless V F
C host wireless V F
D open stations V F

2 Dire se le seguenti affermazioni sono vere o false.
La sigla SNR significa:

A Software No Risk V F
B Signal to Noise Ratio V F
C Signal Not Received V F
D System Nested Root V F

3 Dire se le seguenti affermazioni sono vere o false.
Lo standard di riferimento delle reti wireless prende il nome di:

A 802.11 V F
B 802_11 V F
C 802/11 V F
D 802-11 V F

4 Dire se le seguenti affermazioni sono vere o false.
Quando un host si sposta dall'area di copertura di una stazione base a un'altra:

A la stazione trasmittente riceve un segnale di errore V F
B la comunicazione si interrompe V F
C l'infrastruttura rimodula l'header del frame V F
D cambia il punto di connessione verso l'infrastruttura V F

5 Dire se le seguenti affermazioni sono vere o false.
Quando un segnale attraversa un mezzo fisico si ha:

A distillazione del segnale V F
B deterioramento del segnale V F

C concentrazione dei frame V F
D riduzione del tasso di errore V F

6 **Dire se le seguenti affermazioni sono vere o false.**
Le varie versioni dello standard wireless noto come WiFi condividono:
A la stessa struttura dei frame a livello di connessione V F
B sempre le frequenze su cui operano V F
C il tipo di cablaggio (fibra ottica) per collegare le unità mobili V F
D lo stesso protocollo di accesso V F

7 **Dire se le seguenti affermazioni sono vere o false.**
La sigla BSS significa:
A Business System Survey V F
B Basic Service Set V F
C Broadband Storage Sink V F
D Bold Story Stage V F

8 **Dire se le seguenti affermazioni sono vere o false.**
Per scambiare dati a livello rete, ogni stazione wireless deve:
A essere certificata da Telecom V F
B essere abilitata da ENEL V F
C avere un indirizzo IP che inizia con 128 V F
D essere associata ad un Access Point V F

9 **Dire se le seguenti affermazioni sono vere o false.**
L'associazione con un particolare AP avviene:
A automaticamente appena si rileva la portante V F
B al momento dell'invio del primo ack V F
C dopo aver stabilito la connessione a livello fisico e a livello rete V F
D ogni volta che si accende l'host V F

10 **Dire se le seguenti affermazioni sono vere o false.**
Il frame dello standard 802.11:
A contiene un campo, detto "controllo", molto complesso V F
B coincide con il campo detto "controllo" V F
C non contiene campi di tipo "indirizzo" V F
D contiene il MAC address della stazione che invia il frame V F

11 **Dire se le seguenti affermazioni sono vere o false.**
Nello standard WiFi, i frame di segnalazione sono anche detti:
A beacon frame V F
B bacon frame V F
C backup frame V F
D block frame V F

12 **Dire se le seguenti affermazioni sono vere o false.**
Le seguenti sono denominazioni di standard delle reti cellulari:
A 2G, LTE V F
B GTE, 3G V F
C 8G, BTE V F
D 3G, 4G V F

Sezione **B**

Computare, calcolare, comunicare

1 Teoria della computabilità

Questo capitolo introduce i concetti che rappresentano gli aspetti fondamentali della Teoria della Computabilità.

Partendo da una disciplina antichissima, la logica, questa branca della matematica ha avuto nel XX secolo uno sviluppo straordinario, grazie alla nascita dell'elaborazione automatica: idee e concetti che erano stati spesso proposti a livello astratto hanno trovato applicazioni molteplici, soprattutto se consideriamo l'Informatica nella sua accezione più appropriata di Scienza dell'Informazione.

Dalle comunicazioni ai linguaggi di programmazione, dall'analisi dell'efficienza dell'hardware a quella delle prestazioni del software e degli algoritmi, essa si è rivelata uno strumento indispensabile di progettazione e di verifica.

1 Calcolare e computare

La nascita e lo straordinario sviluppo della Scienza dell'Informazione hanno messo a disposizione degli utenti strumenti con capacità di calcolo un tempo inimmaginabili: anche un semplice Personal Computer, che ormai si trova nelle case di tutti, è in grado di svolgere, grazie al suo hardware ed alla programmabilità, un numero immenso di operazioni elementari nell'unità di tempo.



Figura **1**
Il risultato visivo di un'operazione di rendering 3D del progetto di un edificio.

Calcoli che un tempo avrebbero richiesto stuoli di incaricati, impegnati per settimane o mesi, oltre che per eseguire la sequenza richiesta, anche per verificare di non aver commesso errori, sono ora impostabili in breve tempo su qualsiasi foglio elettronico, o sono a portata di codice per chiunque abbia le necessarie basi di programmazione.

I programmi di grafica, in particolare quelli che offrono il trattamento 3D delle immagini e il rendering del loro aspetto visuale, ottengono risultati di assoluta aderenza al mondo reale grazie ad un numero di operazioni incredibilmente alto, attraverso le quali sono in grado di determinare le proprietà e l'aspetto di ogni punto dell'oggetto virtuale rappresentato.

È intuitivo che al crescere della velocità dei processori, del loro numero e dell'efficienza - dovuta alla loro capacità di operare in parallelo - un calcolo o una sequenza di operazioni possono essere svolti in un tempo sempre più ridotto, fino a diventare, in alcuni casi, quasi istantanei. In realtà la loro durata è sempre finita e non può essere nulla, indipendentemente dalla percezione dell'osservatore, il quale vede solo l'esito finale dell'elaborazione.

Proprio questa grande capacità di svolgere operazioni in tempi estremamente brevi, al di sotto dei tempi di risposta fisiologici degli utenti umani, ha permesso lo sviluppo di applicazioni in grado di eseguire un enorme numero di operazioni in un tempo brevissimo. Si

pensi, ad esempio, ai pacchetti di simulazione in cui si ha la sensazione di essere ai comandi di un aereo supersonico o di una vettura di Formula 1! Un sistema di questo tipo risponde alle azioni dell'utilizzatore con un **feedback** (la risposta del sistema) che si genera con la stessa velocità e con lo stesso contenuto di informazione che l'utente sperimenterebbe se fosse stato veramente ai comandi del dispositivo simulato.



Figura **2**
Flight simulator (simulatore di volo).

Non solo potenza di calcolo • Da quanto detto, potremmo dedurre che lo svolgimento di un qualsiasi calcolo dipenda solo dalla velocità e dal numero dei processori, oltre che dalla qualità del codice utilizzato.

Esistono tuttavia dei limiti, che si possono determinare anche in via teorica, a ciò che un computer può e non può calcolare. Questa problematica è l'oggetto di una disciplina, la **Teoria della Computabilità**, che si è sviluppata a partire dal secolo scorso, la quale si occupa di determinare quanto possa - o non possa - essere affrontato in termini di puro calcolo. Chi si occupa di Informatica, deve conoscerne almeno i principi fondamentali, per essere in grado di prevedere se l'approccio ad un problema complesso, sotto determinate condizioni, potrà o meno trovare soluzione per via numerica.

Logica matematica • La branca della matematica che ha lo scopo di affrontare e studiare gli insiemi, le proprietà logiche degli stessi e delle operazioni che vengono su di esse compiute, prende il nome di **logica formale** o **logica matematica**. Essa tratta i fondamenti del ragionamento e dell'utilizzo di concetti quali *vero*, *falso*, *decidibile*, *corretto*, che sono alla base di qualsiasi trattazione scientifica.

Gli Antichi conoscevano la **Logica**, ma la trattavano da un punto di vista sostanzialmente filosofico: dobbiamo giungere al XIX secolo perché essa assuma, in matematica, l'importanza che oggi riveste. La Logica si è rivelata fondamentale sia nello sviluppo dell'Informatica come scienza dell'elaborazione, sia nell'analisi teorica e generalizzata dei linguaggi, in particolare quelli artificiali, come i linguaggi di programmazione.

Vero e falso, corretto e scorretto • In matematica, il concetto di *correttezza* è fondamentale. Ad esso possiamo pensare in termini di *verità* e *falsità* di una proposizione. Se una proposizione è vera, quanto stiamo affermando è corretto: in caso contrario, non lo è.

Un esempio di espressione aritmetica formalmente corretta, ma falsa, è la seguente:

```
2 + 3 = 7
```

Infatti, 2, 3, 7, + e = sono simboli consentiti dall'aritmetica, e la posizione in cui si trovano nell'espressione sopra riportata è plausibile. L'espressione, però, risulta falsa in quanto, per la definizione del segno =, le quantità a sinistra, ad operazioni svolte, devono essere uguali a quelle a destra, cosa che nel nostro caso non succede.

Se avessimo scritto

```
2 + 3 = 5
```

la proposizione sarebbe stata formalmente corretta e anche vera.

Un caso ancora diverso sarebbe stato il seguente:

```
2 + - 3 7 ==
```

Qui ci troviamo di fronte ad una proposizione che utilizza simboli consentiti dell'aritmetica, ma non rispetta le regole con le quali essi vengono utilizzati. In questo caso parliamo di una proposizione *formalmente* non corretta. In merito alla domanda se essa sia vera o falsa, è facile comprendere che se una proposizione non è formalmente corretta, non ha neppure senso parlare della sua verità o falsità.

Semplificando, possiamo dire che *la logica formale stabilisce le regole che consentono di definire la correttezza e la veridicità delle proposizioni* nel momento in cui si analizzano o si costruiscono.

2 Cosa è computabile?

Chiunque sa che con un semplice PC è possibile svolgere un numero enorme di operazioni in un tempo brevissimo: se ne potrebbe quindi dedurre che, con una adeguata potenza di calcolo, molta memoria e molto tempo a disposizione, un elaboratore possa portare a termine qualsiasi calcolo.

Qualcosa di computabile • È intuitivo che il nostro elaboratore, anche se vecchio e poco efficiente, possa benissimo calcolare il risultato di operazioni come

```
print(8.47 * 9.12)
```

oppure

```
print(13267 / 12.314)
```

o anche

```
print(89234726539 * 39875567000)
```

Potremo avere dei problemi di rappresentazione del risultato, che verrà probabilmente mostrato in notazione esponenziale, ma otterremo il risultato cercato.

L'esperienza ci mostra che, al crescere delle cifre trattate e del numero e del tipo di operazioni presenti nella formula, il tempo impiegato da un medesimo sistema andrà aumentando.

Tempo di calcolo • Dopo aver impostato la linea di comando in modalità interattiva, se chiediamo al PC di eseguire un'operazione banale, come la seguente (redatta in pseudo-C):

```
main()
{
/* programma semplicissimo per eseguire un ciclo di for e */
/* far stampare l'ultimo valore calcolato */

int n;
float i;
float vettore[100];

/* impostazione del numero di iterazioni */

n = 100;
for(i = 1; i <= n; i = i+1)
     {
     vettore[i] = (i/(i+1));
```

```
      }
/* stampo il valore dell'ultimo elemento del vettore */
printf("\nUltimo elemento calcolato %f",vettore[n]);
}
```

il risultato compare praticamente nel momento stesso in cui diamo il comando di Invio.

Se avessimo però richiesto una elaborazione più complessa, come la seguente:

```
/* listato algoritmo più complesso con un numero elevato di cicli */
/* di for e di operazioni e la stampa dell'ultimo valore calcolato */
main()
{

int n;
int i;
float vettore[10000];
/* impostiamo numero di iterazioni *(
n = 10000;

/* all'interno del for assegniamo a vettore[i] un valore che deriva */
/* da una serie di calcoli - naturalmente è solo un esempio */

for(i=1; i <= n;i = i +1)
      {
            vettore[i] = ((i*i/2)/(i*i - 3*i +4));
            vettore[i] = vettore[i] * vettore[i];
      }

/* si noti che il risultato sara' in notazione esponenziale */

printf("l'ultimo elemento trovato vale %f", vettore[n]);

}
```

avremmo dovuto attendere qualche secondo per vedere comparire il risultato: il tempo di attesa, infatti, è determinato dalla potenza del processore a nostra disposizione.

Definizioni per l'approccio algoritmico • Dato un generico algoritmo A usato per risolvere il problema X, notiamo che esiste una relazione che lega il tempo T necessario per risolvere un'istanza di ordine n del problema X usando l'algoritmo A.

Tale relazione può essere sintetizzata nel seguente modo:

$$T = f(n, X, A)$$

Un esempio semplice ci permette di comprendere meglio. Se vogliamo calcolare il tempo T necessario per svolgere un'operazione elementare, quale il prodotto di due numeri interi ognuno dei quali abbia rappresentazione binaria su n bit, possiamo fare alcune considerazioni:

- saranno necessarie al massimo n moltiplicazioni su n bit + n addizioni per gli eventuali riporti per ogni riga
- saranno necessarie al massimo n addizioni per colonna

Quanto visto, ci porta a studiare gli algoritmi sotto un profilo diverso da quanto sinora visto.

Analisi di un algoritmo • Finora, di fronte ad un problema complesso, la nostra principale preoccupazione era di riuscire a definire un algoritmo "funzionante", cioè una sequenza di azioni che, eseguite opportunamente, fornisse in un numero finito di passi il risultato corretto.

La Teoria della Computabilità non si accontenta di un *qualsiasi* algoritmo, purché funzionante, ma, attraverso l'analisi della procedura, si pone la seguente domanda:

"*Dati X ed A, quanto vale T(n, X, A)?*"

Notiamo come, per la definizione di algoritmo, sia necessario che il problema trovi soluzione in un numero finito di passi: il che equivale a dire che lo svolgimento dei calcoli attraverso l'algoritmo *A* deve avvenire in un tempo necessariamente finito.

Gli algoritmi non sono tutti equivalenti: per risolvere un determinato problema possiamo usare metodi alternativi, che comporteranno, dal punto di vista del calcolo, approcci differenti e un numero di operazioni talvolta anche significativamente diverso.

Tra teoria e pratica • Se il tipo di calcolo che dobbiamo eseguire è molto complesso, il tempo *T* può diventare tanto grande da non essere di fatto accettabile. Ad esempio, se il tempo per registrare nell'anagrafica di un ospedale un paziente che si presenta al pronto soccorso fosse di qualche ora, l'utilità del sistema informativo sarebbe praticamente nulla. Anche un'attesa di qualche minuto, in questi casi, può risultare inaccettabile.

In altri ambiti, come nel calcolo strutturale di un edificio in costruzione, sono tollerati anche tempi di calcolo lunghi: in una situazione di questo tipo, il tempo necessario per l'elaborazione non è un fattore critico.

Progettazione di un algoritmo • Se l'efficienza di un algoritmo - in termini di velocità di esecuzione e quindi di tempo di raggiungimento della soluzione - può diventare un fattore decisivo, ci rendiamo conto del perché, in molti ambiti, sia indispensabile imporre determinate prestazioni alle procedure di calcolo.

Come, per la definizione di un algoritmo, il programmatore deve rivolgersi ad un analista, così, in casi come questi, l'analista può avere la necessità - se non è un esperto in materia - di ricorrere a un *progettista di algoritmi*, cioè a una figura in grado di ottimizzare la sequenza di operazioni garantendo che il risultato soddisfi i vincoli temporali richiesti.

Il compito di un progettista di algoritmi è quindi quello di rispondere alla seguente domanda:

"*Dato un particolare X, qual è A che minimizza T(n, X, A) per tutti i valori di n?*"

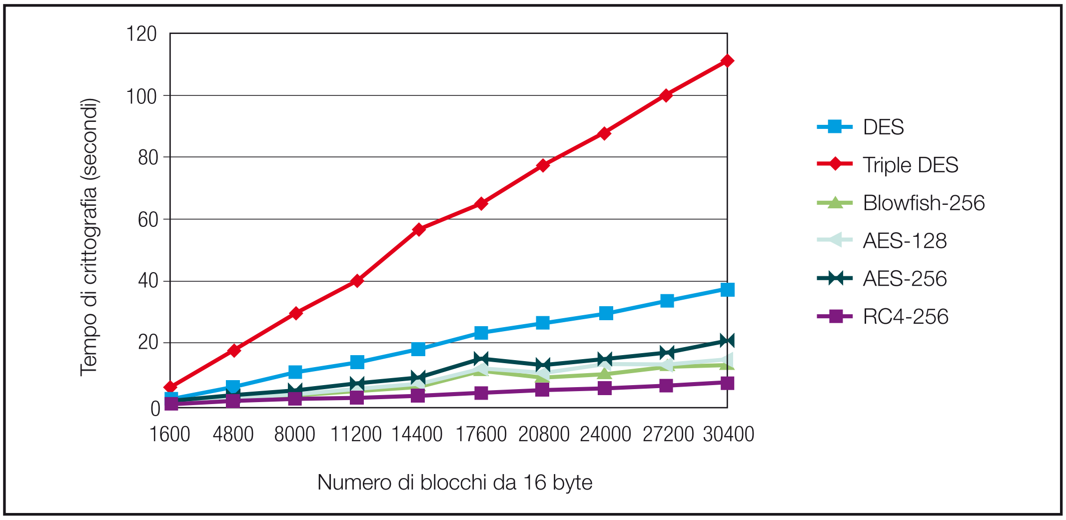


Figura **3**
Tempi di calcolo per uno stesso problema con diversi algoritmi.

Domande fondamentali • Fin dal secolo scorso, i matematici si sono posti una serie di domande, che possono essere così riassunte:

- cosa significa computare?
- cosa può essere computato?
- un computer, se avesse abbastanza memoria e tempo, potrebbe risolvere qualsiasi problema?

Questi quesiti sono tra loro correlati, e rimandano ai temi fondanti della Teoria della Computabilità. Ad essi si collegano anche i seguenti, che fanno riferimento agli aspetti più tecnici:

- quanto velocemente si può risolvere un dato problema?
- quanta memoria sarà necessaria?
- avendo a disposizione una certa quantità di memoria (e solo quella), quali problemi potranno essere risolti e quali no?

Trovare una risposta a queste domande è l'obiettivo della **Teoria della Complessità**, una branca della più generale *Teoria della computabilità*.

Qualcosa di non computabile • Non esiste un metodo universale, cioè che possa essere applicato da un computer e quindi risolto attraverso la scrittura di un codice idoneo, che possa predire se un dato codice non banale - ad esempio in C - potrà funzionare comunque oppure se esistono casi in cui andrà in errore, per esempio entrando in un loop infinito.

Non è necessario affrontare algoritmi intricati per trovarci di fronte a numeri che possono raggiungere ordini di grandezza enormemente grandi come enormemente piccoli. Nella leggenda riportata qui di seguito, la richiesta del matematico Sissa sembra modestissima. Invece - complici le proprietà delle potenze del 2 - quello che sembrava una tazza di chicchi di grano si scopre essere una quantità gigantesca, maggiore di quanto tutti i granai del mondo potrebbero mai contenere.

Sissa e la scacchiera

Un'antica leggenda indiana narra che quando il matematico Sissa inventò il gioco degli scacchi, ne fece omaggio al suo Re. Si presentò con una scacchiera e con i relativi pezzi e gli descrisse le regole e gli scopi del gioco.

Il Re ne fu entusiasta. Ritenendo Sissa meritevole di un premio, lo esortò a chiedere in dono qualsiasi cosa volesse.

Sissa ragionò per un attimo e rispose al Re: "O mio Re, dammi un chicco di grano per il primo quadrato della scacchiera, due per il secondo, quattro per il terzo, otto per il quarto, e così via fino a quando avrai terminato la scacchiera".

Sulle prime il Re si adirò, pensando che chiedere un premio così modesto fosse un affronto alla sua persona: tuttavia, visto che Sissa rimaneva fermo nella sua richiesta, ordinò al tesoriere di accontentarlo.

Passarono le ore, e il tesoriere non si faceva vivo. Il Re lo fece chiamare e gli chiese la ragione del suo ritardo. Il tesoriere gli rispose che gli era impossibile esaudire il sovrano: neppure tutti i granai del mondo avrebbero potuto contenere i chicchi di grano necessari a soddisfare la sua richiesta!

Questa leggenda ha un fondamento reale? Quanti chicchi sarebbero serviti per soddisfare la richiesta di Sissa?

Facciamo il conto: uno (cioè 2^0) nella prima casella; due nella seconda (cioè 2^1); quattro nella terza (cioè 2^2), e così via. Dato che la scacchiera ha N = 64 caselle, il numero di chicchi necessari si può esprimere così:

$$2^{N-1} + 2^{N-2} + 2^{N-3} + \dots + 1$$

Se provassimo a fare il calcolo, troveremmo un numero enormemente grande, anche se non infinito: il solo primo termine della somma vale 9.223.372.036.854.780.000!

3 Ricorsività

Affrontando la Teoria della Computabilità, anche se solo nei suoi aspetti di base, dovremo introdurre temi e definizioni che potrebbero sembrare, a una prima analisi, non pertinenti all'Informatica come la conosciamo e con cui abbiamo a che fare ogni giorno.

Scopriremo in seguito che, in realtà, proprio gli strumenti più potenti, ad esempio i compilatori, non potrebbero essere costruiti con le elevatissime prestazioni richieste dai sistemi attuali, senza una trattazione teorica adeguata, basata proprio sugli elementi della Computabilità.

In particolare, scopriremo che esiste un legame tra le *funzioni ricorsive* e le *funzioni calcolabili*.

Cosa vuol dire ricorsivo? • Nel linguaggio quotidiano, definiamo *ricorsivo* un evento che continua a riproporsi. In ambito informatico, invece, il termine ricorsivo ha un significato diverso: *si definisce ricorsivo ogni algoritmo che viene definito in termini di se stesso*, nel senso che la sua applicazione ai dati oggetto dell'elaborazione si basa sulla suddivisione per semplificazione dell'insieme dei dati stessi e sull'applicazione del medesimo algoritmo all'insieme di dati ridotto.

Vediamolo con un esempio. Supponiamo di voler scrivere un programma che, dato un numero intero *n* in input, ci restituisca il valore dell'ennesimo numero pari.

Ipotizziamo che per noi il primo numero pari sia 0. Quindi, se *n* valesse 3, il secondo numero pari sarebbe 2 e il terzo 4.

Ecco come appare il programma:

```
function pari()
{
     int n;
     int val_cercato;
     if(n==1)
     {
          val_cercato=0;
     }
     else
     {
          val_cercato=pari(n-1)+2
     }
return val_cercato;
}
```

Cosa c'è di particolare?

Notiamo che all'interno della funzione `pari()` viene richiamata la funzione stessa, cui viene passato da elaborare il valore immediatamente precedente, cioè `n-1`.

Se n valesse 3, alla prima iterazione `val_cercato` cercherebbe quindi di calcolare `pari(n-1)`, cioè `pari(2)`. Questo richiamerebbe a sua volta un `pari(1)` che darebbe `0 + 2` come risultato di `val_cercato`, passato alla precedente iterazione, che calcolerebbe `pari(2)` avendo come risultato `2 + 2`, cioè il valore 4 che corrisponde al risultato.

Le funzioni ricorsive sono materia difficile e poco intuitiva: spesso, all'interno del codice sviluppato, permettono di scrivere algoritmi eleganti e molto compatti, ma estremamente difficili da collaudare.

In altri termini, per poter essere eseguito, l'algoritmo richiama se stesso *iterativamente*.

Ricordiamo dalla definizione di algoritmo che l'esecuzione della procedura algoritmica, per quanto complessa, deve comunque avvenire entro un numero finito di passi: nel caso di algoritmi ricorsivi, l'algoritmo termina quando si verifica la cosiddetta funzione di terminazione, tipicamente correlata ad un dato insieme dei valori dei dati in ingresso.

La successione di Fibonacci • Il concetto di ricorsività non è del tutto intuitivo: eppure era noto sin dall'antichità. Un tipico caso in cui lo si incontra è quello della cosiddetta **successione di Fibonacci**, dal nome del matematico Leonardo Fibonacci.

Si tratta di una successione che permette di calcolare il **fattoriale** di un numero che, come è noto dalla matematica, per un certo numero n può essere espresso come:

```
fattoriale(n): n! = 1 * 2 * ... * n
```

Ad esempio:

```
5! = 1 * 2 * 3 * 4 * 5 = 120

7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040
```

Si noti la definizione di fattoriale applicata al valore zero della successione degli interi:

```
0! = 1
```

Possiamo quindi generalizzare la definizione della successione di Fibonacci come segue:

```
0! = 1
n! = 1 * 2 * ... * n-1 * n
```

Esaminiamo l'ultima espressione, osservando che può essere riscritta nella seguente forma:

```
n! = (1 * 2 * ... * n-1) * n
```

ma la porzione tra le parentesi, per definizione di fattoriale, altro non è che:

```
(1 * 2 * ... * n-1) = (n-1)!
```

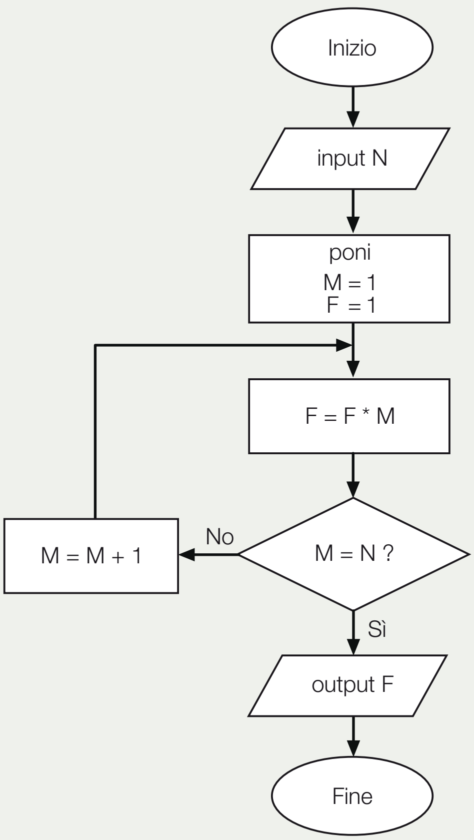
Questo ci porta a poter esprimere `n!` come:

```
n! = (n-1)! * n
```



Figura **4**
Leonardo Fibonacci (1170 - 1250), detto anche Leonardo Pisano, è uno dei fondatori della moderna scienza del calcolo.

Figura **5**
Il diagramma di flusso per il calcolo del fattoriale.



In pseudo-C, la funzione per il calcolo del fattoriale può essere così calcolata:

```
#codice con una funzione recursiva per il calcolo di un fattoriale

# dichiarazione della funzione fattoriale
int  fattoriale(int);

main()
{
     int numero;
     printf("Inserisci un numero intero positivo \n");
     gets(numero);
     if(numero < 0)
          printf("Per cortesia un numero positivo\n");
     else
          printf("Il fattoriale di %d e' %d\n", numero, fattoriale(numero));
}

# codice della funzione ricorsiva, riceve in ingresso numero
function fattoriale(numero)
{
     int kappa;
     if(numero <= 1)
          return 1;
     kappa = numero * fattoriale(numero - 1);
     return kappa;
}
```

Notiamo come, dal punto di vista formale, la funzione risulti estremamente compatta, e come dall'interno di se stessa richiami se stessa: proprio questo aspetto è ciò che ci permette di definirla come *ricorsiva*. La condizione di terminazione viene raggiunta quando `n = 0`.

Condizioni generali • Una volta identificate le funzioni ricorsive, i matematici si sono posti il quesito più generale: se tutti gli algoritmi, opportunamente formalizzati, possano essere ricondotti a funzioni ricorsive. Come l'intuito ci suggerisce, questo non è possibile.

Affinché si possa parlare di ricorsività, è necessario siano contemporaneamente soddisfatte tre condizioni:

1) l'algoritmo deve poter essere espresso in funzione di se stesso
2) deve sempre esistere una condizione di terminazione: in altri termini, non si deve mai presentare il caso che il ciclo di esecuzione prosegua all'infinito (i cosiddetti **loop**)
3) l'algoritmo deve convergere, cioè avvicinarsi progressivamente al valore limite della soluzione.

Compattezza del codice • Uno dei motivi per cui, soprattutto agli albori dell'Informatica, i teorici apprezzarono particolarmente le funzioni ricorsive fu proprio l'estrema compattezza del codice con cui possono essere realizzate. D'altro canto, se non correttamente progettate o verificate, esse possono dare luogo a due tipi di inconvenienti: primo, possono determinare enormi sovraccarichi in termini di risorse di calcolo. Secondo, possono non incontrare la condizione di terminazione, causando un errore dovuto a una eccessiva occupazione di memoria o al superamento dei limiti interni dello **stack**.

Un altro problema che si può presentare è quello dell'efficienza, in termini temporali, nell'esecuzione del codice. Spesso, infatti, la versione ricorsiva è più lenta di altre soluzioni, magari meno eleganti formalmente, ma più veloci nel convergere alla soluzione cercata.

4 Alfabeti e infiniti

Agli inizi degli anni '30 del XX secolo, Alonzo Church (1903-1995), un matematico statunitense, formulò la seguente tesi:

Se una funzione f è ricorsiva, allora essa è computabile ed è computabile attraverso un algoritmo.

La portata di questa affermazione è fondamentale, perché, come possiamo osservare, essa lega biunivocamente il concetto di funzioni ricorsive con quello di funzioni computabili. Dal nostro punto di vista, essere computabile implica essere risolubile da un elaboratore, potenza di calcolo a parte. Notiamo come questo ragionamento sia totalmente indipendente dal linguaggio con il quale l'algoritmo risolutivo è codificato.

Insieme computabile • Andiamo ora a definire quando un insieme sia computabile. Un insieme `A` è computabile quando `XA` è computabile, dove `XA` è così definita:

```
XA(x) = {1 se x ∈ A, 0 altrimenti}
```

Questa definizione non si applica solo agli insiemi di dati ma anche, più genericamente, agli insiemi di relazioni. Ai nostri fini, però, non approfondiremo oltre.

Alfabeto • Nell'ambito della logica formale, prende il nome di **alfabeto** un insieme Σ (*sigma*) - finito e non vuoto - di caratteri o di simboli sui quali possiamo operare secondo regole. I simboli che compongono un alfabeto, in questo contesto, sono anche detti **lettere**. Esiste anche l'**elemento nullo** dell'alfabeto, denotato con ε.

Vediamo qualche esempio.

Un alfabeto composto da un numero finito di caratteri è dato da:

```
a, b, c, d
```

Mentre l'alfabeto così definito:

```
0, 1, 2, ... , N
```

comprendente l'insieme dei numeri interi, ha infiniti termini.

Giustapposizione • Consideriamo l'operazione che ci permette di concatenare parole costruite a partire da un alfabeto: essa prende il nome di **giustapposizione**. Ad esempio, se il nostro alfabeto è:

```
{a, b, c, d, e, f, g}
```

e abbiamo due parole

`w1 = aba` e `w2 = bbc`
possiamo giustapporle come `w1 . w2`, ottenendo

```
ababbc
```

o come `w2 . w1`, ottenendo

```
bbcaba
```

Notiamo che la parola-risultato appartiene ancora all'alfabeto di partenza.

Gli alfabeti hanno solitamente un numero limitato di componenti; ne esistono però alcuni che ne hanno un numero illimitato.

Introduciamo allora un nuovo concetto, relativo alla *numerosità* dell'insieme di caratteri che compone un alfabeto: la *cardinalità*.

Cardinalità di un insieme • Si dice **cardinalità** di un insieme finito il numero dei suoi elementi. Ad esempio, l'insieme `(a, b, k, w, z)` ha cardinalità 5, mentre l'insieme `(0, 1, 2, ... , 10)` ha cardinalità 11.

Insiemi infiniti • Se il numero dei componenti dell'insieme è infinito, le cose si complicano. Non tutti gli infiniti sono equivalenti: per esempio, è infinita la sequenza dei numeri naturali:

```
0, 1, 2, ... n
```

Un insieme come questo viene comunemente classificato come insieme *N (speciale)* e la sua cardinalità viene indicata con il termine $\aleph_0$ (si legge "alef zero"), in cui il termine $\aleph$ è la prima lettera dell'alfabeto ebraico.

Ogni insieme infinito in cui ogni elemento possa essere posto in corrispondenza biunivoca con l'insieme dei numeri naturali ha cardinalità $\aleph_0$.

Anche l'insieme dei numeri interi, dato da:

```
-n, -n +1, ..., -1, 0, +1, +2, ... , n
```

viene indicato come *Z speciale* e ha cardinalità $\aleph_0$.

Se consideriamo, invece, l'insieme dei numeri irrazionali, notiamo come essi non possano essere posti in corrispondenza biunivoca con l'insieme dei numeri naturali (infatti sono più numerosi). La loro cardinalità viene espressa con un infinito di ordine superiore allo 0: diremo, infatti, che essa vale $\aleph_1$.

Le grandezze infinite che identifichiamo attraverso il simbolo $\aleph$ ($\aleph_0$, $\aleph_1$, ecc.) prendono il nome di **numeri transfiniti**.

Stringhe • Dato un insieme Σ finito e non vuoto del tipo appena visto, cioè un alfabeto, diamo il nome di **stringa** (o **parola**) a ogni **ennupla** (significa *disposizione finita*) di caratteri di Σ.

Lunghezza di una stringa • La lunghezza della stringa, come è intuitivo, corrisponde al numero di caratteri che la compongono e viene indicata con il simbolo `|w|`.

Se una stringa ha lunghezza zero, essa viene indicata con ε (si legge *epsilon*) o λ *(si legge lambda)*.

L'alfabeto binario • Un alfabeto semplicissimo è quello binario. Per questo alfabeto sarà:

$$\Sigma = \{0,1\}$$

Esempi di stringhe che possiamo ottenere con i componenti di questo alfabeto sono i seguenti:

$$\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$$

L'insieme di tutte le stringhe ottenibili a partire da un alfabeto Σ viene indicato con Σ*.

Un insieme di stringhe su Σ, che corrisponde per definizione ad un sottoinsieme di Σ*, prende allora il nome di *linguaggio formale su* Σ* *e* ad esso vengono associate, dal punto di vista matematico, importanti proprietà.

5 Linguaggi formali

Possiamo definire come *linguaggio L su un certo alfabeto* (se esso comprende un numero finito di termini) l'insieme di tutte le parole costruibili a partire da tali termini.

Se però l'alfabeto comprende un numero infinito di termini, non è possibile applicare la definizione appena vista: in questi casi la definizione di L dovrà comprendere una proprietà $P(w)$ che risulti vera solo su tutte le parole di L.

Per rappresentare linguaggi infiniti, abbiamo sostanzialmente due diversi approcci: il primo prende il nome di *approccio generativo*, il secondo di *approccio riconoscitivo*.

Approccio generativo • Questo tipo di definizione si basa sull'identificazione di una regola, o procedura, in grado di generare, attraverso un automatismo, tutte le parole di un linguaggio.

Approccio riconoscitivo • In questo caso, la descrizione di un linguaggio L avviene attraverso l'identificazione di un algoritmo, che indichiamo con A, tale per cui, se forniamo in input una certa parola w ad A, l'output sarà 1 (cioè vero) se la parola appartiene al linguaggio, 0 in caso contrario.

Se per un linguaggio L siamo in grado di costruire un riconoscitore, allora esso potrà anche essere costruito per via generativa.
Non è invece vero che esista sempre un riconoscitore per qualsiasi linguaggio generato.

Un esempio di approccio generativo • Un tipico esempio di approccio generativo è dato dalle cosiddette **grammatiche**. Ai nostri fini possiamo tranquillamente usare la grammatica della nostra lingua naturale, l'italiano, proprio nell'accezione in cui abbiamo iniziato a considerarla fin dai primi anni di scuola.

Se consideriamo le seguenti regole:

- diamo il nome di *frase* alla giustapposizione di un soggetto, di un predicato e di un complemento
- diamo il nome di *soggetto* alla giustapposizione di un articolo e di un nome
- diamo il nome di *complemento* alla giustapposizione di un articolo e di un nome
- un possibile articolo è "*la*"
- un possibile nome è "*mucca*", oppure "*erba*" oppure "*acqua*"
- un possibile predicato è "*mangia*" oppure "*beve*"

rispettando questa grammatica, possiamo costruire le seguenti frasi:

la mucca mangia l'erba

la mucca beve l'acqua

Il nostro alfabeto di riferimento è dato da:

{la, mucca, erba, acqua, mangia, beve}

Le regole che applichiamo possono essere viste formalmente come:

frase	(costituita da soggetto + predicato + complemento)
soggetto	(costituita da articolo + nome)
articolo	(la)
nome	(mucca / erba)
predicato	(mangia / beve)

Metasimboli e simboli terminali • Concetti come *predicato*, *soggetto*, *frase* servono per costruire frasi a partire da un certo alfabeto, ma non sono parte dell'alfabeto stesso: ad essi facciamo riferimento in termini di **metasimboli** o **simboli non terminali**, mentre i termini che fanno parte dell'alfabeto sono detti **simboli terminali**.

Si noti che è possibile rappresentare le proposizioni appena viste anche sotto forma di albero binario:

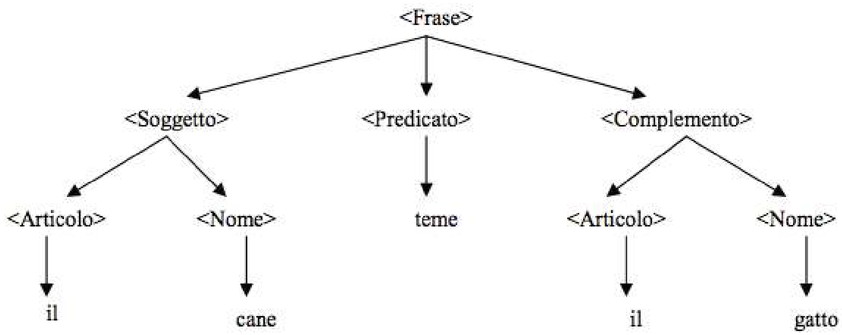


Figura **6**
La proposizione "il cane teme il gatto" nella rappresentazione ad albero binario.

Questa rappresentazione appena esplicita chiaramente il concetto di *simbolo terminale* e di *simbolo non terminale*.

Concetti essenziali

- Esistono dei limiti a quello che un computer può e non può calcolare.
- Un computer, per potente che sia, ha una memoria finita: perciò l'elaborazione di un algoritmo deve terminare in un tempo finito.
- Non tutti gli algoritmi che risolvono uno stesso problema sono equivalenti in termini di efficienza.
- L'efficienza di un algoritmo è correlata alla sua velocità di esecuzione.
- Un algoritmo ricorsivo è un algoritmo definito in termini di se stesso, che richiama se stesso durante la propria applicazione.
- Se una funzione è ricorsiva, allora è computabile ed è computabile attraverso un algoritmo.
- In logica formale, un alfabeto è un insieme non vuoto di simboli o caratteri sui quali si può operare secondo determinate regole.
- Si dice cardinalità di un insieme finito il numero dei suoi elementi.
- Anche per gli insiemi infiniti si definisce una cardinalità, che viene denotata con $\aleph$.
- Il simbolo matematico $\aleph$ indica un numero transfinito.
- Dato un alfabeto, è possibile definire uno o più linguaggi su tale alfabeto.
- Per rappresentare linguaggi infiniti, possiamo usare un approccio generativo o un approccio riconoscitivo.

Test

1 **Dire se le seguenti affermazioni sono vere o false.**
Se avessimo a disposizione molto tempo e un'adeguata quantità di risorse hardware, un elaboratore:

A potrebbe autoprogettarsi V F
B non potrebbe risolvere comunque alcune classi di problemi V F
C non sarebbe mai in grado di completare l'esecuzione di un codice scritto in C++ V F
D sarebbe sempre meno veloce di una procedura di calcolo manuale V F

2 **Dire se le seguenti affermazioni sono vere o false.**
La logica matematica prende anche il nome di:

A logica formale V F
B logistica formale V F
C logicità formale V F
D logorrea formale V F

3 **Dire se le seguenti affermazioni sono vere o false.**

La Teoria della Computabilità è:

A una disciplina che studia i sistemi di traduzione automatica da una lingua all'altra V F

B un'antica scienza greca riscoperta nel sec. XIX V F

C è una branca della neurologia V F

D la disciplina che studia cosa un computer può e non può computare V F

4 **Dire se le seguenti affermazioni sono vere o false.**

L'espressione x +/- 3 = y è:

A valida solo se x = 4 e y = 5 V F

B calcolabile solo per valori dispari di x V F

C logicamente informe V F

D formalmente scorretta V F

5 **Dire se le seguenti affermazioni sono vere o false.**

Un algoritmo si dice ricorsivo se:

A viene definito in termini di se stesso V F

B non converge mai a una soluzione finale V F

C risolve qualsiasi tipo di problema V F

D non è riconducibile a un codice eseguibile V F

6 **Dire se le seguenti affermazioni sono vere o false.**

Tutti gli algoritmi:

A sono intercambiabili V F

B risolvono un dato problema nello stesso tempo V F

C sono equivalenti dal punto di vista dell'efficienza V F

D possono essere analizzati in termini di correttezza e tempo di esecuzione V F

7 **Indicare quali delle seguenti affermazioni (una o più di una) sono vere.**

I linguaggi formali possono essere di tipo:

A generativo V F

B riconoscibile V F

C generalizzato V F

D riconoscitivo V F

Esercizi

1 **Si considerino le seguenti espressioni e si commentino dal punto di vista logico, in termini di verità/falsità e di numero delle soluzioni:**

A a = 30b

B = a + 30b

C 13k + 27 = 40

D x - 12 = x - 11

2 **Si analizzi il seguente codice e si determini se esso termina correttamente la propria esecuzione:**

```
main()
{
float n;
float m;
int i;

i = 20
m = 0
n = 8

/* qui svolgiamo le operazioni necessarie per
arrivare al calcolo voluto */

if(i<50)
          {
          m = m * i;
          n = n*m;
          i = i + 3;
    }
```

3 **Si determini la cardinalità dei seguenti insiemi:**

A insieme dei numeri pari positivi <= 50

B insieme dei numeri dispari multipli di 4

C insieme dei numeri negativi multipli di 3

D insieme dei numeri interi il cui valore è 14/2

4 **Si analizzi il seguente codice e si determini se esso termina correttamente la propria esecuzione:**

```
main()
{
float n;
float m;
int i;

i = 20
m = 0
n = 8

/* qui svolgiamo le operazioni necessarie per
arrivare al calcolo voluto */

if(i<50)
    {
            m = m * i;
            I = i -1;
            n = n*m;
            i = i + 3;
            m = m +i;
            i = i - 2;
    }

}
```

2 Teoria degli automi

"Automazione" ed "automatico" sono termini divenuti familiari, nel XX secolo, anche al vasto pubblico: in questo capitolo, affronteremo il concetto basilare di automa, nell'accezione che esso assume nel mondo matematico e nel contesto della Scienza dell'Informazione.

Introdurremo il formalismo grafico con cui in tutto il mondo queste entità vengono trattate: è una rappresentazione con caratteristiche particolarmente semplici. Grazie a essa, si possono descrivere molteplici realtà e modellizzare situazioni e problemi correlati al mondo informatico, sia per ottenere generalizzazioni formali, come nel caso dei linguaggi di programmazione, sia per fornire strumenti di progettazione e simulazione, ad esempio in ambito circuitale.

1 Il concetto di automa

Il termine **automa** indica, in senso generale, qualcosa che è in grado di funzionare in modo *automatico*, cioè, essenzialmente, senza essere azionato da un agente esterno. Esempi di automi sono i **robot**: quelli più semplici compiono sequenze di azioni seguendo procedure prefissate; i modelli più sofisticati sono anche in grado di reagire in modo diverso a situazioni che i loro sensori percepiscono come differenti.

In questo capitolo introduciamo il concetto matematico di *automa* e forniamo alcuni concetti di base di quella che va sotto il nome di **Teoria degli Automi**.

La Teoria degli Automi è una branca della Matematica.

Cos'è un automa? • ***Un automa è una macchina a stati, cioè un sistema virtuale che può esistere solo in determinate configurazioni - dette, appunto, stati - e che per passare da uno stato all'altro necessita di uno specifico evento.***

Questa è una definizione semplificata e non rigorosa, ma ci permette di introdurre il tema, che poi analizzeremo a un livello più approfondito.

Per aiutarci, a livello intuitivo, introdurremo un esempio molto pertinente ad uno dei campi di applicazione degli automi matematici.

Formalismo • La Teoria degli Automi utilizza una serie di convenzioni grafiche per descrivere oggetti e procedure.

Ogni *stato* in cui la macchina si può trovare è indicato da un cerchio: in ogni istante, la macchina si trova in uno e uno solo di questi stati.

Un *evento* in grado di far passare l'automa da uno stato iniziale q_0 a uno stato q_1, diverso dal primo, è rappresentato con un arco orientato che va da q_0 a q_1, in questo modo:

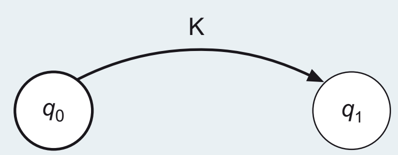


Figura 1

L'evento è connotato da un simbolo o una sigla, cui corrisponde una tabella di descrizione.

Possono esistere eventi che non spostano l'automa dal suo stato corrente: essi vengono indicati con un arco orientato che ritorna sullo stato stesso, come in figura 2.

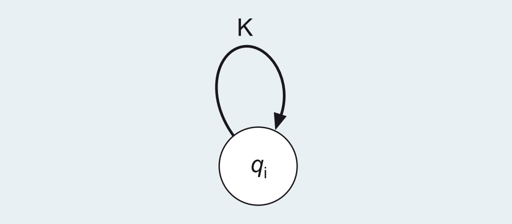


Figura **2**

Nella rappresentazione classica, si indica lo stato iniziale dell'automa con un cerchio a bordo ingrossato e lo stato finale con due cerchi, uno entro l'altro, nel seguente modo:

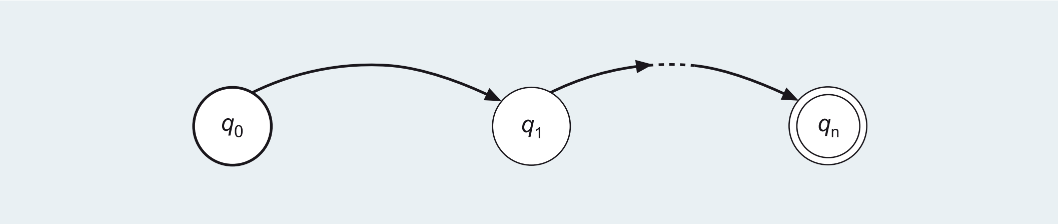


Figura **3**

Grafi orientati • Una rappresentazione grafica come quella descritta prende il nome di **grafo orientato**. In essa, ogni arco è associato a una direzione di percorrenza, che indica, in altri termini, cosa è antecedente e cosa susseguente, e definisce quindi un rapporto che può essere sia di causa ed effetto, sia di priorità in senso temporale.

Stati finiti ed infiniti • Esistono automi che possono essere descritti da un numero finito di stati (cioè possono assumere solo un numero finito, per quanto grande, di configurazioni). Altri automi hanno un numero di stati infinito.

Nel primo caso parliamo di automi a stati finiti, nel secondo di automi a stati infiniti.

Per i nostri scopi, tratteremo solo gli automi a stati finiti.

Determinismo e casualità • Due definizioni che ritroveremo spesso nelle prossime pagine sono quelle di **sistema deterministico** e di **sistema non deterministico**.

Un sistema si dice *deterministico* se, in qualsiasi stato, è perfettamente prevedibile, noti gli ingressi e le condizioni al contorno, quale sarà il suo stato successivo: ciò equivale, in altri termini, a poterne pienamente prevedere il futuro comportamento, noti lo stato, la struttura e gli input.

Un sistema si dice *non deterministico* (o *aleatorio*, o *casuale*) se, anche conoscendo perfettamente il suo stato e gli ingressi, non è possibile prevedere con certezza quale sarà il suo stato futuro.

Una **rete logica** è un esempio classico di sistema deterministico. Se applichiamo due ingressi a "1" a un circuito di *AND*, l'uscita sarà "1"; e avremo sempre - e solo - l'uscita "0" in tutte le altre combinazioni di ingressi.

Esempi di sistemi non deterministici sono il comportamento del corpo elettorale in occasione di una votazione, oppure quello dei consumatori di fronte a scelte di acquisto tra prodotti diversi. Per descrivere i comportamenti di questo tipo, infatti, si rendono necessari i metodi della statistica, come avevamo illustrato quando abbiamo affrontato il tema della modellizzazione dei problemi.

2 Automi deterministici a stati finiti

Per capire cosa è e come funziona un ***automa deterministico a stati finiti***, possiamo immaginarlo come uno speciale computer il cui input sia costituito da un nastro di carta sul quale

è tracciata una sequenza di simboli. In particolare, i simboli sul nastro sono caratteri, proprio come le stringhe di codifica di un linguaggio interpretato. Ogni simbolo - carattere che la macchina legge, la fa passare in uno stato diverso da quello in cui si trova. Questo processo continua finché si presenta una delle due seguenti situazioni:

- la sequenza di simboli, che la macchina ha letto uno dopo l'altro, la manda in uno stato detto di ***accettazione della sequenza*** (o di *riconoscimento della sequenza*)
- la sequenza di simboli, che la macchina ha letto uno dopo l'altro, la manda in uno stato detto di ***non accettazione della sequenza*** (o di *non riconoscimento* o di *rigetto*) della sequenza.

Facciamo un esempio. Supponiamo che il nostro automa stia analizzando i comandi di un linguaggio molto semplice, che comprende istruzioni come `PRINT`, `SOMMA`, `SOTTRAI`.
Se sul nastro sta scritto: `PRONT`, l'automa percorrerà la seguente sequenza:

- legge `P` e lo accetta perché è il primo carattere del comando `PRINT` che è compreso nel suo linguaggio
- dopo `P`, non essendoci altri comandi che iniziano per `P` tranne `PRINT`, l'unico carattere che può accettare è `R`. Infatti legge `R` e lo accetta.
- a questo punto l'unico simbolo accettabile è `I`. Invece si trova di fronte il carattere `O`, che non viene accettato, e la sequenza termina con un rifiuto del comando in input.

Rivediamo l'esempio in una rappresentazione di tipo grafico:

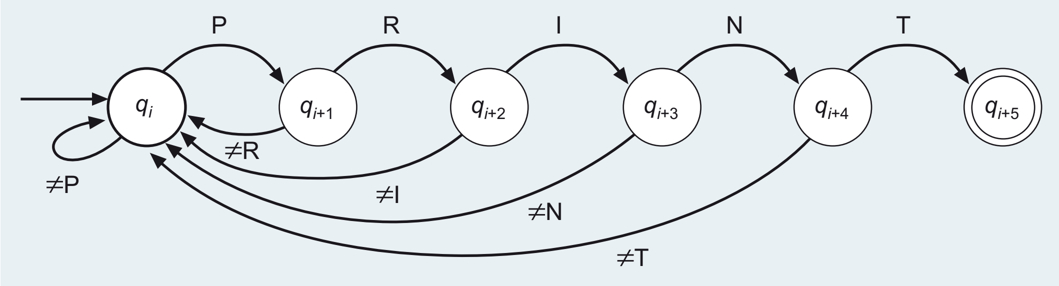


Figura **4**

Una descrizione intuitiva

• Possiamo descrivere un ***automa deterministico a stati finiti*** come una macchina composta da un numero finito n di stati, collegati tra loro da archi le cui etichette sono simboli che provengono da un alfabeto finito.
Ad ogni lettura di un simbolo, la macchina cambia stato. Il nuovo stato è funzione esclusivamente dello stato del sistema al momento dell'input e degli archi etichettati che fuoriescono da esso.

Ad esempio, questo automa:

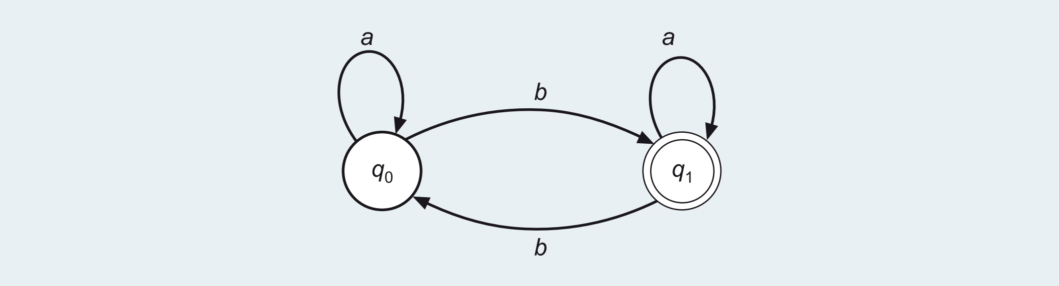


Figura **5**

ha due soli stati possibili: q_0 e q_1.

Le etichette dei suoi archi orientati sono *a* e *b*. Diciamo, quindi, che il suo alfabeto Σ è dato da:

$$\Sigma = \{a,b\}$$

Un automa come questo riesce a portarsi stabilmente nello stato finale q_1 se - e solo se - la stringa di input presenta un numero dispari di caratteri *b* consecutivi.

Vediamo cosa succede, per esempio, se in input avessimo la sequenza *abaab*:

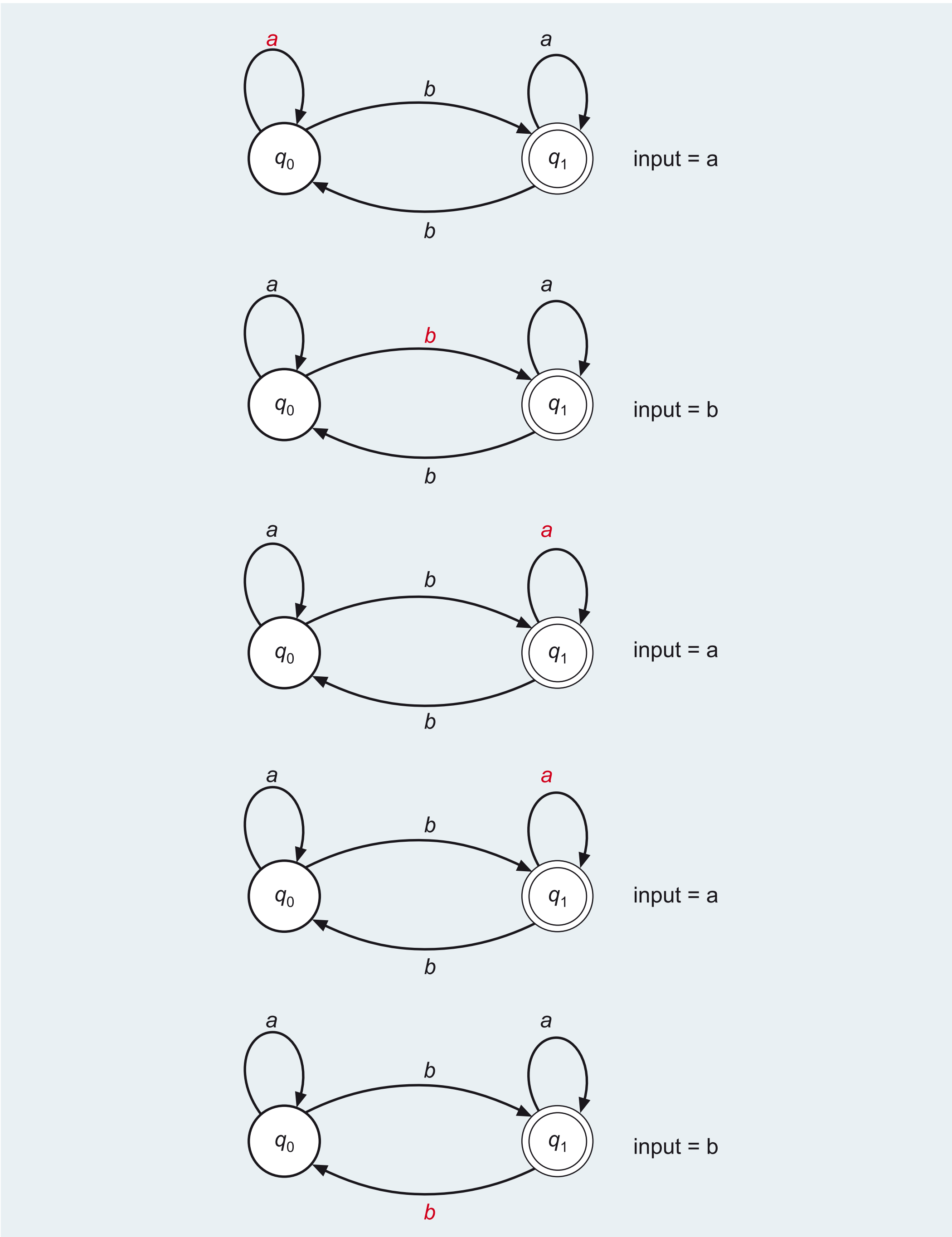


Figura **6**

Al termine della stringa ci ritroviamo in q_0, che non è uno stato finale, o di riconoscimento, accettato.

Se la nostra stringa fosse stata, invece, la sequenza *abbb*, avremmo avuto:

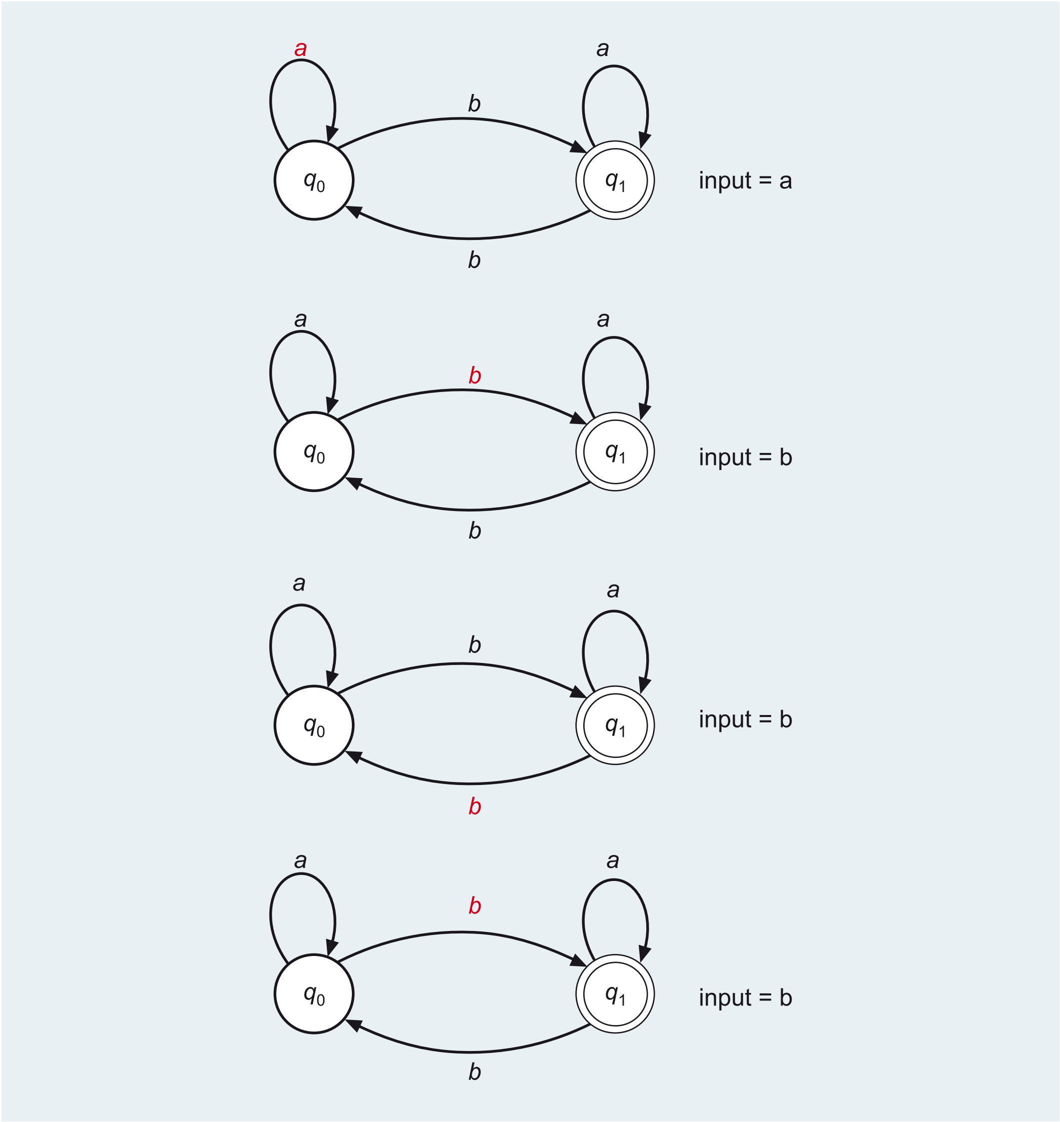


Figura **7**

È facile vedere che al termine della stringa l'automa si trova nello stato q_1, che è quello di terminazione: la sequenza in ingresso è stata accettata.

Definizione formale • Una possibile definizione formale di *automa deterministico a stati finiti* è la seguente:

Si dice automa deterministico a stati finiti una quintupla {K, Σ, q_0, F, δ} in cui K è un numero finito di stati, Σ un alfabeto finito, $q_0 \in K$ è uno stato iniziale definito e unico, e δ è una funzione che porta da K × Σ a K.

In ogni momento, la situazione di un automa di questo tipo può essere compiutamente descritta da una tripla *(x, q, y)* in cui *x* è la porzione di nastro che è già stata letta, *q* lo stato corrente dell'automa e *y* la porzione di nastro ancora da leggere.

Un aspetto interessante di questo tipo di automi è quello di poterci permettere di prevedere in modo rigoroso se in ciascuno dei 1, 2, ..., *n* passi il risultato previsto sarà o non sarà ottenuto. In particolare, ci interessa quello che succede in un solo passo: formalmente, possiamo

dire che un automa M deterministico a stati finiti, ad esempio $M(K, \Sigma, \delta, q_0, F)$, se si trova in:

$$M(x, q, y)$$

genererà la situazione

$$(x', q', y')$$

in un solo passo se e solo se

esiste un simbolo $\delta \in \Sigma$ tale per cui la macchina legga un solo simbolo

e

$$\delta(q,\sigma) = q'$$

cioè il carattere letto σ mandi da q a q'.

3 Automi non deterministici a stati finiti

Affrontiamo ora l'altra famiglia di automi a stati finiti: quelli di tipo *non deterministico*.

Proviamo ad immaginare un automa descritto da un grafo di questo tipo:

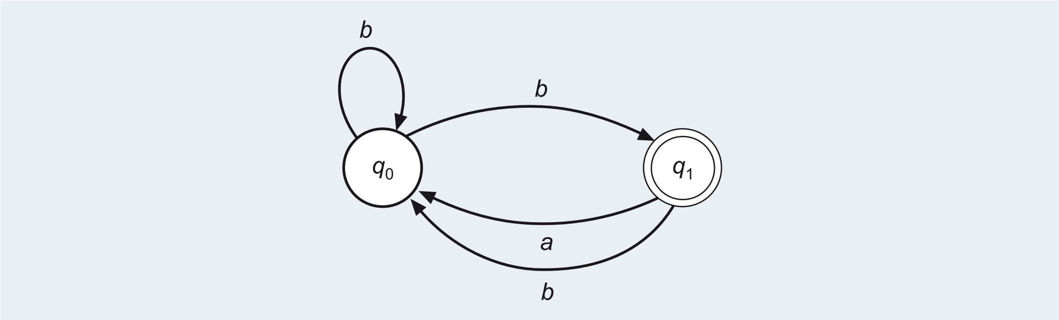


Figura 8

Se ci troviamo nello stato iniziale q_0 e sul nastro di input leggessimo b, il sistema dove si porterebbe?

In realtà non lo possiamo predire: possiamo solo dire che *potrebbe* andare in q_1 ma anche restare in q_0, nulla di più.

Quando l'automa si trova in q_1, qualsiasi sia il carattere letto, a oppure b (ricordiamo che stiamo considerando un alfabeto $\Sigma = \{a,b\}$) il sistema ritornerà in q_0.

Caratteristiche • Gli automi di questo tipo presentano delle caratteristiche interessanti. In particolare:

- gli archi orientati possono essere associati alla stringa nulla ε
- possono esistere più archi associati alla stessa etichetta fuoriuscenti da un medesimo stato

In altri termini, possono avvenire passaggi di stato anche se sul nastro incontriamo la stringa nulla (cioè non stiamo leggendo alcun carattere).

Finora, ogni *evento* era stato il risultato di un'azione ben precisa: la lettura di un carattere sul nastro. Negli automi di tipo *non deterministico*, può essere che il sistema si porti in un nuovo stato anche senza avere letto nulla.

Negli automi deterministici, ogni cambiamento di stato era legato alla funzione δ, cioè a un evento ben preciso. Per i non deterministici non possiamo più parlare di passaggi di stato legati a δ: perciò, per poter esprimere questa capacità del sistema di evolvere, dobbiamo introdurre in luogo di una precisa funzione un concetto più ampio, quello di relazione, che indicheremo con Δ.

Definizione formale • Possiamo quindi dare una definizione formale di questo tipo di sistemi:

Un automa non deterministico a stati finiti M è una quintupla (K, Σ, Δ, q_0, F) in cui K, Σ, q_0 ed F hanno lo stesso significato specifico degli automi deterministici, mentre Δ è una funzione di relazione di transizione.

Un esempio interessante, che utilizza il piccolo alfabeto Σ {*a*,*b*}, è quello di un sistema che genera un linguaggio in cui le frasi, cioè le stringhe riconosciute, abbiano le seguenti caratteristiche:

- la stringa deve iniziare con *b*
- se all'interno della stringa abbiamo il carattere *a*, esso deve essere preceduto e seguito da una *b*

La rappresentazione di questo automa è la seguente:

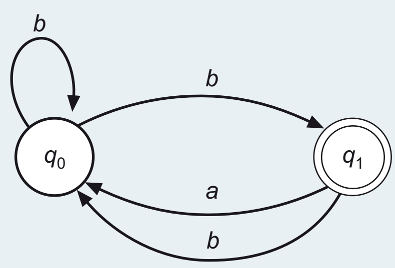


Figura **9**

Notiamo che da q_0 escono, associati all'evento *b*, due archi: siamo di fronte ad un sistema non deterministico. Quando il nastro presenta il carattere *b*, non possiamo a priori prevedere se il sistema passerà allo stato q_1 o resterà nello stato iniziale q_0.
Sotto quali condizioni, allora, una sequenza sarà accettata da un automa di questo tipo?

Nel caso di automi non deterministici a stati finiti, una sequenza di caratteri in ingresso è riconosciuta se esiste almeno un percorso in cui, partendo da q_0, si termini a q_1.

Senza volerci addentrare in dettagli che sono al di fuori della portata di questo corso, prendiamo come un dato di fatto la seguente affermazione:

Gli automi deterministici a stati finiti generano gli stessi linguaggi che possono essere generati dagli automi non deterministici a stati finiti.

Una considerazione intuitiva è che gli automi deterministici sono un sottoinsieme di quelli non deterministici, caratterizzati però da un numero minore di gradi di libertà.

4 Linguaggi regolari

Ora che abbiamo introdotto le necessarie definizioni, passiamo all'applicazione degli automi all'analisi ed alla sintesi di linguaggi, intesi in senso informatico e matematico.

Iniziamo con la definizione di **linguaggio regolare**. Si tratta di un concetto interessante e abbastanza particolare: scopriremo che può essere generato da un automa a stati finiti e che accetta una grammatica di tipo regolare.

Per un linguaggio regolare sono valide le seguenti asserzioni:

1) l'insieme vuoto è un linguaggio regolare
2) per ogni stringa $x \in \Sigma^*$, $\{x\}$ è un linguaggio regolare
3) se A e B sono linguaggi regolari, allora $A \cup B$ è un linguaggio regolare
4) se A e B sono linguaggi regolari, allora AB è un linguaggio regolare
5) se A è un linguaggio regolare, allora lo è anche A^*
6) nient'altro può essere considerato un linguaggio regolare

In queste asserzioni incontriamo il simbolo di asterisco (*), il cui significato verrà chiarito nei paragrafi che seguono.

Analisi degli enunciati • Chiariamo ora il significato delle asserzioni.

La (1) è un'affermazione che viene posta come vera per definizione.

La condizione (2)

"*Per ogni stringa* $x \in \Sigma^*$, $\{x\}$ *è un linguaggio regolare*"

significa che ognuno dei simboli dell'alfabeto Σ può essere un linguaggio in sé e per sé, ma lo è anche una qualsiasi concatenazione di simboli dello stesso alfabeto.

La condizione (3)

"*Se A e B sono linguaggi regolari, allora* $A \cup B$ *è un linguaggio regolare*"

significa che se L_1 è un linguaggio regolare, il cui alfabeto è $\{a,b\}$, e se è regolare anche L_2, il cui alfabeto è $\{c,d\}$, l'alfabeto di riferimento di L_3, unione logica di L_1 e L_2 sarà regolare:

$$L_3 = L_1 \cup L_2 = \{a, b, c, d\}$$

La condizione (4)

"*Se A e B sono linguaggi regolari, allora AB è un linguaggio regolare*"

afferma la regolarità del linguaggio $L_3 = L_1L_2$, cioè concatenazione di due linguaggi regolari L_1 e L_2.

L'asterisco che appare accanto ad alcuni simboli, ad esempio nel termine A^* della condizione (5), prende il nome di **stella di Kleene**. Esso segnala che una proprietà vera per un elemento dell'insieme resta valida anche per ripetizioni - comunque lunghe - dello stesso elemento.

Se un linguaggio L_1 ha come alfabeto $\{a,b\}$, allora anche

$$L2 = L*1$$

cioè un linguaggio che ha per alfabeto $\{a, aa, aaa, ..., b, bb, bbb...\}$ è un linguaggio regolare.

Corrispondenza • Lo studio di queste proprietà permette di enunciare la seguente affermazione:

L'insieme dei linguaggi regolari coincide con quello dei linguaggi che possono essere accettati da un automa a stati finiti.

Un altro modo di dire la stessa cosa è:

Un insieme di stringhe costituisce un linguaggio di un automa a stati finiti se - e solo se - esso costituisce un linguaggio regolare.

Nelle righe che precedono, abbiamo richiamato il concetto di **grammatica regolare**: ricordiamo che essa può essere di tipo *generativo* o *analitico*.

Grammatica generativa • Una **grammatica generativa** parte da un simbolo dell'alfabeto e costruisce tutte le possibili frasi determinate a partire da quel simbolo, secondo le regole caratteristiche della grammatica stessa.

Grammatica analitica • Una grammatica analitica può essere vista come un insieme di regole in grado di analizzare una qualsiasi stringa in ingresso o - come si dice talvolta - cioè di ridurla in porzioni collegate da operatori logici, del tipo AND, OR, NOT, e di determinare se essa è o non è compresa nella lingua descritta dalla grammatica.

Un buon esempio di grammatica analitica è costituito dai **parser**. Si tratta di analizzatori lessicali, cioè programmi che prendono in carico le linee di codice del software e stabiliscono se ognuna di esse è formalmente corretta: se tutti i termini della linea di codice sono dichiarati, usati e correlati correttamente - secondo le regole del linguaggio di programmazione usato - la linea di codice verrà riconosciuta come valida e passata all'esecuzione; in caso contrario, verrà rigettata con indicazioni - più o meno dettagliate - circa gli errori incontrati.

Descrivere o analizzare? • Per semplificare, possiamo immaginare una grammatica analitica come uno strumento che insegna come leggere una lingua, mentre una grammatica generativa insegna come scriverla, fornendo le regole per combinarne gli elementi in modo corretto.

Grammatiche regolari • Le grammatiche regolari vengono definite anche come *grammatiche sensibili al contesto*, in contrapposizione alle grammatiche dette *context free*, o libere dal contesto.

In una grammatica *regolare*, la parte sinistra di una **regola produttiva**, cioè di una regola che permette di costruire in senso logico una frase correttamente, può essere solo un simbolo non terminale, con i seguenti vincoli:

- tale simbolo può essere il simbolo nullo (indicato con ε)

oppure

- un simbolo terminale seguito da un solo simbolo non terminale

Grammatiche libere dal contesto • Prendono il nome di *grammatiche libere dal contesto* quelle grammatiche in cui la parte sinistra di una regola produttiva può essere solo formata da un unico simbolo non terminale.

Una grammatica libera dal contesto si dice anche regolare se, alla sinistra della regola produttiva, si trova un'espressione che contiene un solo simbolo non terminale.

Altre grammatiche • Esistono altri tipi di grammatiche, che non vengono presi in considerazione in questa sede.

5 Applicazioni in ambito informatico

Visto l'elevato livello di astrazione delle definizioni incontrate nelle pagine precedenti, coloro che si occupano di elaboratori e di Scienza dell'Informazione potrebbero chiedersi quale sia l'interesse ad approfondire una materia così complessa in termini tanto lontani dal contesto pratico ed operativo.

La risposta è questa: quanto introdotto - se pure a livello di base - permette di affrontare il tema generale dei linguaggi, che non solo trova applicazione nell'ambito della linguistica, ma che ha a che fare con quel tipo particolarissimo di linguaggi che sono i linguaggi artificiali per la programmazione, come C, R, BASIC, ecc.

I linguaggi formali - e la loro definizione attraverso gli automi a stati finiti - permettono di formalizzare delle grammatiche che possono essere viste come un insieme di regole per generare delle stringhe in modo corretto.

Gli automi, d'altro canto, sono un'eccellente rappresentazione di quelle attività, tipiche di ogni elaboratore come vedremo nel prossimo capitolo, di riconoscimento di stringhe e di loro interpretazione, nella trasformazione di un comando comprensibile ad un umano in una sequenza di 0 e di 1 comprensibili per un processore.

Nel loro insieme, ci permettono di affrontare in modo formale e scientifico, ed in senso lato, il tema dei linguaggi, siano essi di tipo naturale oppure di tipo artificiale, come nel caso di quelli tipici della programmazione degli elaboratori.

Nel mondo reale • L'analisi degli automi a stati finiti ci consente di studiare le limitazioni teoriche e intrinseche di un linguaggio e, in generale, della computazione.

Lo studio di questi argomenti da parte dei teorici della Scienza dell'Informazione ha portato alla scoperta del fatto che esistono problemi che non possono essere risolti in un tempo finito, e per i quali quindi nessun elaboratore, per quanto potente, potrà mai fornire una soluzione certa.

Gli automi a stati finiti sono inoltre diventati uno strumento fondamentale per la definizione delle componenti di un compilatore o di un interprete di un qualsiasi linguaggio.

In questi ambiti è infatti indispensabile l'esistenza di una componente che abbia il compito di analizzare lessicalmente le righe di codice, estrarne gli elementi chiave, quali i comandi, le operazioni, i parametri ed i termini su cui comandi ed operazioni vengono applicati.

Risultato primario di tale analisi sarà la verifica della correttezza formale, quindi grammaticale, del testo inserito e la sua accettazione per la trasformazione in comandi interni dell'elaboratore o il suo rigetto, con la conseguente generazione di una indicazione di errore. La loro natura di macchine virtuali ha inoltre trovato anche un'importante applicazione nel settore delle reti di comunicazione.

Concetti essenziali

- Un automa è una macchina a stati, cioè un sistema virtuale che può esistere solo in determinate configurazioni.
- Per passare da uno stato un altro un automa richiede un evento.
- Gli automi vengono rappresentati attraverso il formalismo dei grafi orientati, in cui archi orientati, associati a eventi, congiungono il simbolo di un determinato stato con quello di uno (o più) altri stati.
- Un sistema si dice deterministico se, noti gli ingressi e le condizioni al contorno, è perfettamente prevedibile quale sarà il suo stato successivo.
- Si dice automa deterministico a stati finiti una quintupla $\{K, \Sigma, q_0, F, \delta\}$ in cui K è un numero finito di stati, Σ un alfabeto finito, $q_0 \in K$ uno stato iniziale definito e unico, e δ una funzione che porti da $K \, X \, \Sigma$ a K.
- Un linguaggio regolare può essere generato da un automa a stati finiti ed accetta una grammatica di tipo lineare.
- Un insieme di stringhe costituisce un linguaggio di un automa a stati finiti se - e solo se -esso costituisce un linguaggio regolare.
- Una grammatica regolare può essere di tipo generativo o analitico.

Test

1 Dire se le seguenti affermazioni sono vere o false.
Nel contesto della Scienza dell'Informazione, un automa è:
A un sistema virtuale che può esistere solo in determinate configurazioni V F
B un programma in grado di autocorreggersi V F
C un sistema operativo virtuale che scrive da solo le proprie applicazioni V F
D un elaboratore dotato di volontà propria V F

2 Dire se le seguenti affermazioni sono vere o false.
La rappresentazione canonica degli automi avviene attraverso:
A i diagrammi di flusso V F
B gli insiemi di Eulero-Venn V F
C i grafi orientati V F
D uno pseudocodice dedicato V F

3 Dire se le seguenti affermazioni sono vere o false.
Un sistema si dice deterministico se:
A note le condizioni al contorno e gli input, ne possiamo calcolare sempre la struttura V F
B in qualsiasi stato, nota la sua struttura e i suoi input, il suo comportamento futuro è prevedibile V F
C se applicando qualsiasi ingresso, esso giungerà sempre allo stato finale in un numero finito di passi V F
D il suo stato futuro è sempre prevedibile, indipendentemente dagli input applicati V F

4 Dire se le seguenti affermazioni sono vere o false.
In un automa non deterministico a stati finiti:
A possono esistere più archi associati alla stessa etichetta in uscita da un medesimo stato V F
B ogni stato presenta sempre più archi in uscita associati alla stessa etichetta V F
C non esistono più archi in uscita da un medesimo stato V F
D ogni stato ha lo stesso numero di archi in uscita ed in ingresso V F

5 Dire se le seguenti affermazioni sono vere o false.
Se A e B sono due linguaggi, A ∪ B è un linguaggio regolare se:
A Almeno A è un linguaggio regolare V F
B Solo uno tra A e B è regolare V F
C Sia A che B sono iper-regolari V F
D Sia A che B sono entrambi regolari V F

6 Dire se le seguenti affermazioni sono vere o false.
L'insieme dei linguaggi regolari coincide con quello dei linguaggi che:
A hanno una qualsiasi struttura logica V F
B sono compresi da almeno un'etnia in Europa V F
C possono essere accettati da un automa a stati finiti V F
D trovano rappresentazione in un linguaggio di programmazione V F

7 Dire se le seguenti affermazioni sono vere o false.
I linguaggi formali possono essere definiti attraverso:
A gli automi a stati finiti V F
B gli automi non-deterministici V F
C le grammatiche ipercontestuali V F
D l'analisi semantica del parlato V F

8 **Dire se le seguenti affermazioni sono vere o false.**

Un utilizzo degli automi a stati finiti è:

A lo studio della propensione all'acquisto di prodotti informatici V F

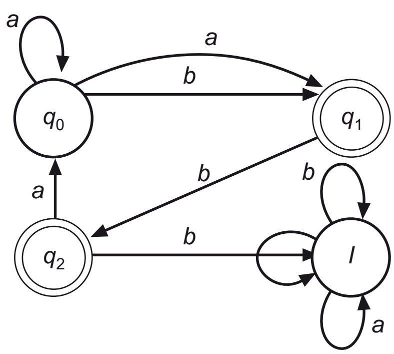
B la costruzione di migliori rapporti interpersonali V F

C la comprensione della struttura psicologica della popolazione V F

D l'analisi lessicale delle righe di codice V F

Esercizi

1 **Si analizzi il seguente automa a stati finiti, in grado di operare su un linguaggio {a,b}:**

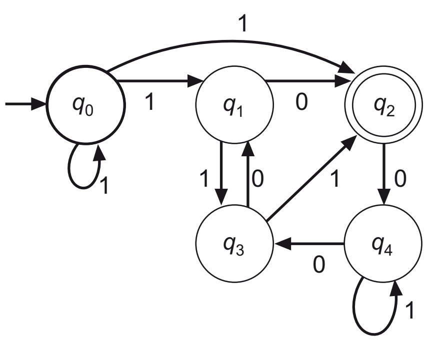


A Cosa succede se la stringa in ingresso è pari a *c*?

B Cosa succede se la stringa in ingresso è pari a *babab*?

C Cosa succede se la stringa in ingresso è pari a *aaaabbb*?

2 **Si analizzi il seguente automa a stati finiti:**



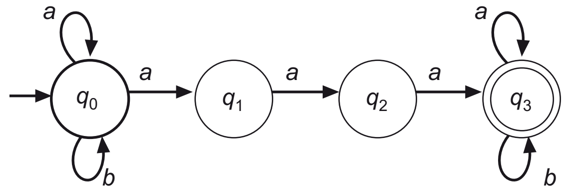
e si dica in quale stato esso si porta a partire da q_0 se gli input prossimi sono pari a:

A 1 0 0 1 0 0 1

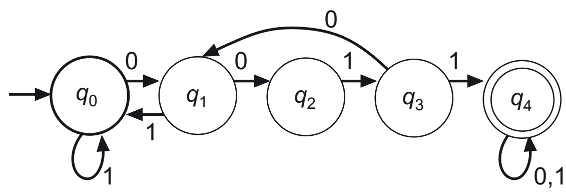
B 0 1 0 1 0 0 0 1

C 1 1 0 0 1 0

3 **Si analizzi il comportamento del seguente automa a stati finiti (riconoscitore) e si determini qual è la sequenza che esso riconoscerà:**



4 **Si analizzi il seguente automa a stati finiti:**



e si dica in quale stato esso si porta a partire da q_2 se gli input prossimi sono pari a:

A 0

B 0 0 0

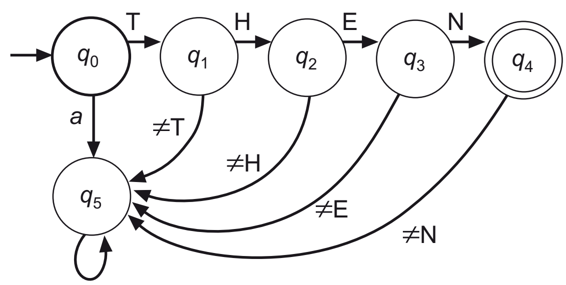
C 0 1 0

5 **Si analizzi il comportamento del seguente automa a stati finiti (riconoscitore) nei confronti delle possibili sequenze in input:**

A THIS

B THAN

C THEN



e si commenti il suo funzionamento, descrivendolo.

6 **Si progettino gli automi a stati finiti in grado di riconoscere le seguenti stringhe:**

A Franco

B 123875

C 91FR23

7 **Si progetti un automa a stati finiti in grado di accettare qualsiasi stringa che non contenga 3 zeri consecutivi.**

3 La macchina e il test di Turing

La Scienza dell'Informazione quale noi la conosciamo deriva, nei suoi concetti fondamentali, dagli studi del britannico Alan Turing negli anni '40. In questo capitolo si introducono tre temi fondamentali della sua opera. Il primo è quello relativo all'analisi dei problemi che sono risolubili con l'aiuto di un calcolatore, cioè che sono "computabili". Il secondo ci presenta una speciale macchina virtuale, la Macchina di Turing, mostrandone le relazioni logiche con il tema della computabilità. Il capitolo si conclude con uno degli argomenti più affascinanti della storia dell'Informatica: il Test di Turing, con il quale vengono poste a confronto le capacità di pensiero umano con quelle dell'elaboratore.

Le implicazioni anche filosofiche di questo tema sono molteplici, e docenti ed allievi potranno quindi trovare, a partire da queste pagine, spunti per ricerche ulteriori ed approfondimenti.

1 Un genio dell'Informatica

In uno dei volumi precedenti di questo testo abbiamo introdotto la figura di Alan Turing, scienziato britannico formatosi come matematico e dedicatosi a diversi campi di applicazione di questa scienza, quale, ad esempio, la decrittazione di codici cifrati.

Il suo genio ne fece una delle figure di spicco nel reperimento della chiave risolutiva del codice segreto utilizzato dai militari tedeschi durante la Seconda Guerra Mondiale.



Figura **1** Alan Turing (1912-1954).

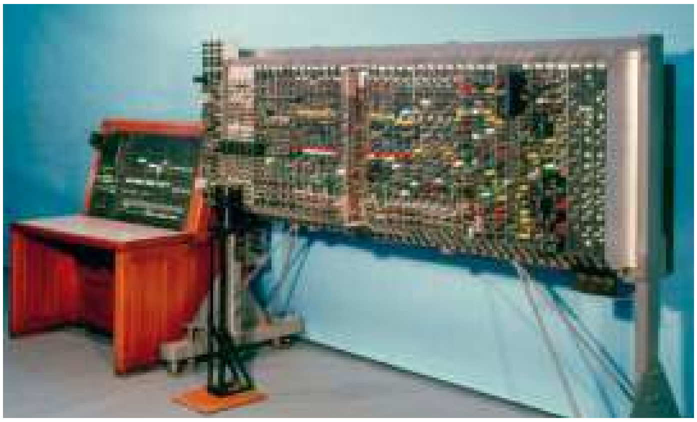


Figura **2** Ricostruzione del computer ACE progettato da Turing nel 1950.

In ambito informatico, la figura di Turing è di straordinaria importanza: intorno al 1945, infatti, giunse a formalizzare le caratteristiche teoriche di un elaboratore, partendo da uno scritto di Von Neumann, ma ampliandone la portata ed il livello di generalità. Nel 1946 descrisse, per la prima volta, in una sua pubblicazione e in modo completo e rigoroso, la struttura di un elaboratore che immagazzinava le istruzioni di programma in una memoria di tipo elettronico.

Tutti i computer che utilizziamo, sia i computer personali (PC), sia gli elaboratori dedicati a compiti speciali, come quelli che fanno funzionare i telefoni cellulari o controllano l'elettronica di bordo di un'automobile, sono sempre strutturalmente riconducibili a questo modello generale.

La fama di Turing è legata a due progetti fondamentali: la "**Macchina**" e il "**Test**" che da lui presero il nome. Li approfondiremo nei prossimi paragrafi. In particolare, con l'analisi del secondo muoveremo i primi passi nell'affascinante disciplina dell'Intelligenza Artificiale.

2 Problemi e soluzioni

Ogni volta che ci poniamo un problema e cerchiamo di trovarne la soluzione, tendiamo a dare per scontato che tale soluzione esista. Nel caso di un programma, scritto in un qualsiasi linguaggio, vorremmo che svolgesse le operazioni necessarie per arrivare al risultato che desideriamo.

A titolo di esempio, supponiamo di avere un codice semplicissimo, scritto in pseudo-C:

```
main()
{
     printf('Buongiorno a tutti!');
}
```

Se non abbiamo compiuto errori di digitazione nella fase di compilazione, quando lo eseguiamo comparirà sul monitor la scritta:

```
Buongiorno a tutti!
```

e il programma terminerà correttamente la propria esecuzione.

Immaginiamo, ora, che il codice da analizzare sia più complesso. Per esempio, quello di un programma che:

- riceva in ingresso un numero intero, diciamo n, maggiore di due.
- verifichi se esistono 3 numeri interi, x, y e z, tali per cui: $x^n + y^n = z^n$
- se tali valori esistono, scriva a monitor:

I valori trovati sono x, y e z

Semplice, vero? Vediamolo in pratica.

Nel caso in cui $n = 3$, l'espressione di cui si cerca la soluzione è

$$x^3 + y^3 = z^3$$

e il senso pratico del nostro problema diviene:
"*Esistono tre numeri naturali x, y e z per cui x al cubo + y al cubo sia uguale a z al cubo?*"

Espresso in questa forma, il problema può apparire facilmente risolubile. Posto invece nei suoi termini generali, ha costituito un vero e proprio scoglio per generazioni di matematici.

Se scrivessimo un programma per risolvere il quesito posto, andrebbe in loop e dovremmo interromperne forzatamente dall'esterno l'esecuzione.

Il teorema più difficile del mondo

"*Esistono tre numeri naturali, x, y e z tali per cui $x^n + y^n = z^n$ per qualsiasi valore di n?*"

Per secoli gli studiosi avevano cercato di trovare una terna di numeri che soddisfacesse questa equazione, ma senza riuscirci: provando con valori di n crescenti, si poteva constatare che l'equazione non aveva soluzioni, ma il calcolo - eseguito a mano - diventava

proibitivo via via che il valore di n aumentava. Nell'ambiente dei matematici, l'idea che non esistesse alcuna soluzione si andava consolidando: si trattava però di dimostrarlo.



Figura **3**
Il matematico francese Pierre de Fermat (1601 - 1665).

Nel XVII secolo, il matematico francese Pierre de Fermat sostenne di avere trovato la dimostrazione. In una nota in margine a una copia di un antico testo di matematica, l'*Aritmetica* del greco Diofanto, egli scrisse di essere in grado di dimostrare che non esistono soluzioni del problema per $n > 2$. Aggiunse anche che non aveva potuto aggiungere alla nota anche la dimostrazione perché il margine del libro era troppo stretto.

La questione rimase aperta, e divenne di dominio comune come la *dimostrazione dell'ultimo teorema di Fermat*. Il teorema fu anche verificato per alcune condizioni isolate (ad esempio per $n = 3$ e per $n = 5$), ma la dimostrazione generale restò irraggiungibile per più di due secoli dopo la morte di Fermat.

Qualche decennio fa, con l'aiuto di un computer, si riuscì a verificare per tentativi che per tutti i valori di $n < 4\,000\,000$, nessun numero naturale soddisfaceva l'equazione del teorema in questione. Si trattava, però, di risultati insufficienti ai fini di una vera dimostrazione: infatti, non si poteva escludere che qualche terna di numeri naturali, per $n > 4\,000\,000$, costituisse una soluzione.

Si dovette giungere al 1994 perché il matematico britannico Andrew Wiles riuscisse a ottenere, con metodi sofisticati, una dimostrazione generale: per coglierne la portata, si pensi che la sua esposizione richiedeva 130 pagine. Ancora oggi, la dimostrazione di Wiles è comunque al di là della comprensione non solo del grande pubblico, ma di gran parte dei matematici.

Programmi di test • Sugli sviluppi dell'ultimo teorema di Fermat, gli studiosi di Informatica si sono posti un quesito di carattere più generale, ma più vicino alle loro problematiche concrete: "*Dato un codice generico P (cioè un programma o una parte di esso), è sempre possibile scrivere un programma H che analizzi P e ci dica se, qualsiasi siano i dati in ingresso, esso terminerà la propria esecuzione correttamente oppure entrerà in loop?*"

In termini pratici il programma *H* avrebbe lo scopo di collaudare, o, come si dice in gergo, di *testare* il codice *P*.

Analizziamo la questione riferendoci a un codice *P* sia del tipo visto poco sopra:

```
main()
{
     printf('Buongiorno a tutti!');
}
```

Ipotizziamo di avere l'ingresso *I* e il programma da testare *P* applicati al medesimo codice di verifica, che indicheremo con `TESTER`.



Figura **4**
Struttura del Tester.

A seconda dell'esito del test, il programma `TESTER` darà in uscita un `SI`, se il programma *P* termina regolarmente la sua esecuzione in un tempo finito; un `NO` in caso contrario.

Si può dimostrare per assurdo come non sia possibile creare un programma `TESTER` che sia in grado di verificare *qualsiasi* codice.

B3-01
Limiti del TESTER

Un esempio • Vediamo un problema semplice (almeno all'apparenza):

```
while x != 1 do
     x = x - 2
```

supponiamo che i dati passati in ingresso al programma (cioè i valori attribuiti a `x`) siano interi positivi.

Per valori dispari di `x`, l'espressione

```
x = x - 2
```

continua a decrementare il valore di `x` fino a che si raggiunge il valore

```
x = 1
```

e il programma termina correttamente la sua esecuzione, dato che la condizione iniziale sul ciclo di `while` non è più soddisfatta.

Se invece il valore `x` in ingresso fosse pari, il test sul ciclo di `while` non incontrerebbe mai il valore 1, che verrebbe saltato in fase di decremento, e il programma continuerebbe all'infinito a sottrarre due dal valore ottenuto nel ciclo precedente.

Se supponiamo di definire una funzione

```
pari(x)
```

che restituisce `TRUE` se il valore `x` è pari, il nostro codice diviene:

```
while x != 1 do {
     if pari(x) then
          x = x/2
     else
          x = 2x + 1
}
```

il programma terminerebbe correttamente solo per quei valori di `x` in ingresso che fossero

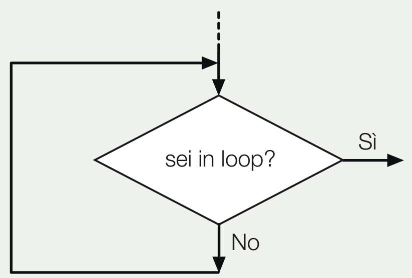


Figura **5**
Loop infinito.

potenze del due. Per tutti gli altri continuerebbe a *ciclare in loop*.

Se, con una piccola modifica, il programma diventasse:

```
while x != 1 do {
     if pari(x) then
          x = x/2
     else
          x = 3x + 1
{
```

il programma terminerebbe correttamente per i valori di `x` pari a potenze del due, ma per altri valori di `x` terminerebbe solo dopo un numero imprevedibile di cicli, talvolta facendo assumere ad `x` valori molto alti e non prevedibili.

Non saremmo quindi in grado, a priori, di stabilire se il programma incontra o meno la condizione di terminazione.

Il problema di terminazione • Questo problema, che prende il nome di problema di **halting** (in italiano **terminazione**), è esprimibile formalmente nel seguente modo:

"*È possibile stabilire a priori se un programma P, dato un ingresso consentito X, terminerà o meno la propria esecuzione in un tempo finito?*"

Si tratta di un classico problema al quale non è possibile dare risposta affermativa.

Non esiste, infatti, in generale, un algoritmo di qualsiasi tipo che possa risolvere il problema dato.

Indecidibilità • Ci troviamo di fronte ad un concetto fondamentale della Teoria della Computabilità: esistono problemi per i quali non c'è soluzione, indipendentemente dalla potenza di calcolo, dalla dimensione della memoria, dal numero di elaboratori e dal tempo che venissero impegnati per cercare tale soluzione.

Questi problemi prendono il nome di **indecidibili**.

3 La Macchina di Turing

Nei capitoli precedenti abbiamo incontrato la cosiddetta **Macchina di Turing**: non si tratta di un elaboratore reale, come è invece il nostro PC, ma solo una *macchina virtuale*, cioè un artificio logico-matematico, il cui comportamento, programmabile, simula una macchina reale.

La sua importanza è riassumibile nella seguente affermazione:

Una funzione è computabile se può essere risolta da una Macchina di Turing.

Si tratta di una proprietà importantissima: dato che, essenzialmente, gli elaboratori di cui disponiamo sono modellizzabili come Macchine di Turing, scopriamo che essi sono in grado, almeno potenzialmente, di affrontare qualsiasi funzione computabile.

La struttura della macchina • Possiamo immaginare una Macchina di Turing come un dispositivo fisico in grado di compiere elaborazioni sui dati in ingresso. Questi dati vengono somministrati in forma di simboli riportati su di un nastro di lunghezza infinita. Il nastro è suddiviso in quadrati di identica dimensione, in ognuno dei quali è riportato un simbolo, cioè un carattere dell'alfabeto (finito) che la macchina è in grado di elaborare. In alternativa, uno o più quadrati (in numero finito) possono anche essere vuoti (in gergo, **blank**).

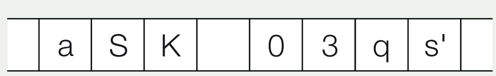


Figura **6**

La Macchina ha una testa in grado di leggere e di scrivere, operando però su un solo quadrato alla volta, più precisamente quello che si trova al di sotto del suo spazio operativo:

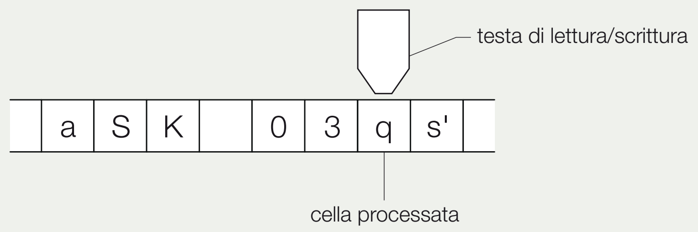


Figura **7**

Un'ulteriore condizione necessaria per definire la Macchina di Turing è che essa, in un qualsiasi istante *t*, si trovi in uno stato q_i che sia uno di *n* possibili stati, in cui *n* è un numero finito.

La Macchina è inoltre descritta da un insieme di istruzioni che assumono la seguente struttura:

```
(stato_corrente, carattere_corrente, nuovo_stato, nuovo_carattere, sini-
stra, destra)
```

Cosa significa? Non è difficile: in un determinato istante la Macchina di Turing, si trova nello stato

```
stato_corrente
```

e con la testa di lettura legge il

```
carattere_corrente
```

quindi passa al

```
nuovo_stato
```

sostituendo al `carattere_corrente` contenuto nel quadrato in corso di elaborazione il

```
nuovo_carattere
```

e spostando la testa di lettura di una posizione sul nastro verso destra (o verso sinistra, a seconda di quanto specificato nell'istruzione).

Una macchina elementare • Per fare un esempio, possiamo immaginare di aver codificato sul nastro in ingresso una serie di numeri interi positivi, scritti in **notazione unaria**. Ricordiamo che in questo tipo di rappresentazione un qualsiasi numero *n* è rappresentato da ($n + 1$) caratteri "`1`" posti uno di seguito all'altro. Per esempio, secondo questa notazione, il numero 4 viene rappresentato come

```
11111
```

e il numero 7 viene rappresentato come

```
11111111
```

Supponiamo anche che il nastro sia letto da sinistra verso destra e che i numeri riportati siano separati gli uni dagli altri da uno e un solo quadrato vuoto (*blank*).

Come possiamo immaginare una Macchina di Turing che sia in grado di eseguire moltiplicazioni sui numeri in ingresso e fornire il risultato?

Immaginiamo che il nastro porti le codifiche dei numeri 3 e 4, cioè abbia questa struttura:

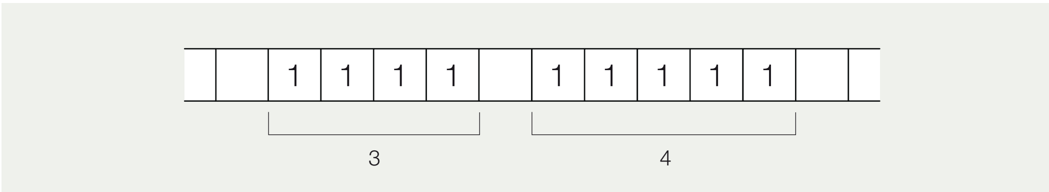


Figura **8**

Otterremo il risultato voluto se, dopo aver eseguito il ciclo delle istruzioni, avremo sul nastro una sequenza di "1" come questa:
e la testa della Macchina si troverà posizionata sul bit più a sinistra della sequenza che costituisce il risultato.

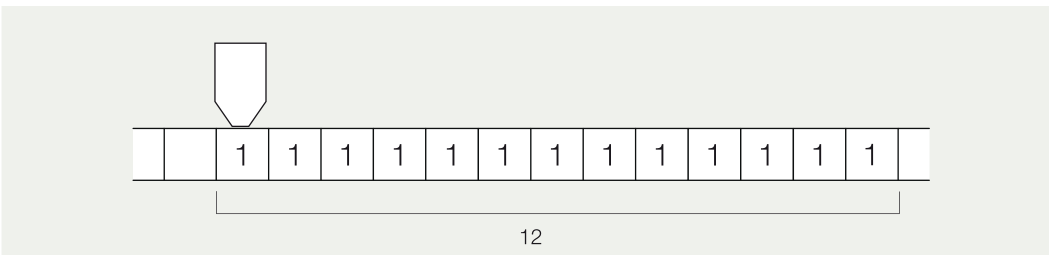


Figura **9**

Un automa a stati finiti • Un modo diverso di descrivere la Macchina di Turing ne evidenzia il profondo legame con le entità che abbiamo analizzato nelle pagine precedenti, gli *automi a stati finiti*.

Ipotizziamo che la Macchina sia in uno stato definito q e che u e v siano due stringhe consentite all'interno dell'alfabeto Σ del nastro in ingresso.

Descriviamo con il termine

$$u\,q\,v$$

una configurazione in cui lo stato corrente sia q, il contenuto corrente del nastro sia $u\,v$ e la testa si trovi sul primo elemento che compone la stringa v. Ad esempio:

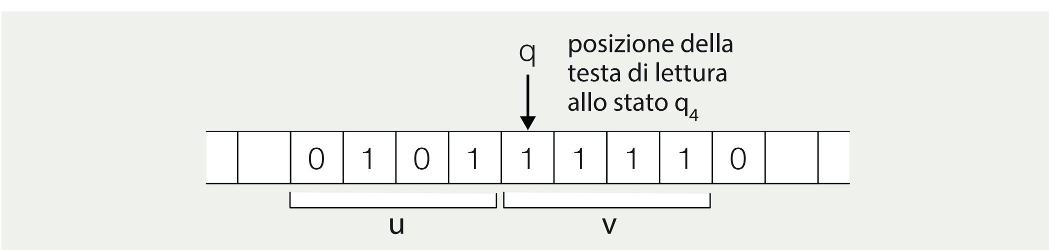


Figura **10**

ci mostra una situazione in cui la sequenza di caratteri sul nastro è data da

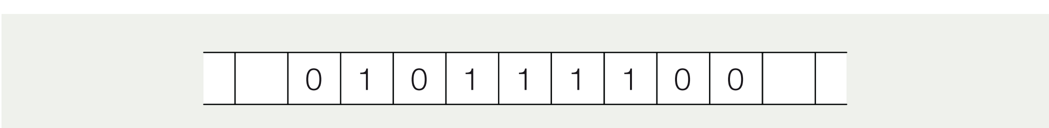


Figura **11**

Lo stato interno è descritto da q_4 e la testa di lettura / scrittura si trova sul terzo "1" del nastro (letto da sinistra verso destra).

Diciamo che una configurazione, descritta come C1, genera una configurazione C2 se la Macchina di Turing può, in modo consentito, andare da C1 a C2 in un solo passo.

Ma la descrizione che abbiamo appena visto ci è familiare: si tratta delle stesse definizioni che abbiamo utilizzato quando abbiamo illustrato gli automi a stati finiti. Una Macchina di Turing, infatti, è a tutti gli effetti un automa a stati finiti.

Stati speciali • Alcuni stati assumono un significato particolare. Uno di essi è lo stato di *accettazione*, cioè la configurazione che la macchina assume quando i dati in ingresso hanno superato le verifiche di congruenza dettate dalla grammatica del linguaggio utilizzato. Il suo opposto è lo stato di *non accettazione* o *rigetto*, nel quale il sistema si va a posizionare se le verifiche sui dati in ingresso hanno dato esito negativo.

A questo punto dovrebbero risultare intuitive le relazioni tra la Macchina di Turing, come ente astratto, e i sistemi di analisi del codice - i parser - utilizzati dai compilatori e dagli interpreti dei linguaggi. Solo se comandi, istruzioni e valori sono strutturati secondo le regole del linguaggio sarà possibile, per il processore, svolgere le azioni contenute nella codifica del listato. Per questo motivo il parser analizza, byte dopo byte, le linee di codice di un programma, comportandosi come una sorta di Macchina di Turing a cui si chiede di stabilire se le istruzioni e i parametri impostati sono o non sono accettabili secondo le regole, quindi la grammatica, del linguaggio adottato.

4 L'Intelligenza Artificiale

Uno dei quesiti che gli scienziati si posero fin dal momento della nascita degli elaboratori fu il seguente:

"***Può una macchina pensare?***"

Per dare una risposta a tale domanda, dobbiamo prima chiarire cosa intendiamo con il termine "pensiero" (inteso nell'accezione di *prodotto dell'azione del pensare*).

Una definizione sfuggente • Ai primordi dell'Informatica, si credette di poter facilmente simulare, grazie agli elaboratori, tutta una serie di azioni che erano ritenute tipiche della mente umana e, quindi, legate alla capacità del nostro cervello di "pensare".

Risultò subito evidente l'abilità dei computer di eseguire calcoli numerici complessi con velocità superiori a quelle degli umani. Invece le cose non andarono altrettanto bene quando le attività cerebrali che si intendevano replicare per via elettronica riguardavano altre capacità, quali, ad esempio, il prendere decisioni sulla base di situazioni non prevedibili a priori, oppure l'analisi di un linguaggio di tipo verbale ed il suo utilizzo: in altre parole l'uso, in input ed in output, di ciò che si definisce un **linguaggio naturale**.

La disciplina che si occupa della possibilità di simulare il comportamento intelligente della mente umana prende il nome di **Intelligenza Artificiale** (indicata con **AI**, per Artificial Intelligence).

Negli anni '80 dello scorso secolo, gli esperti pensarono di essere ad un passo dal creare macchine, cioè robot estremamente sofisticati, in grado di simulare in tutto e per tutto il comportamento di un essere intelligente. Si parlò in quegli anni di *robot di Quinta Generazione*, capaci di comportarsi, anche se solo in ambiti particolari, come un essere pensante, umano o animale.

In quel periodo, per fare un esempio, gli utenti non professionali erano stupiti dalle capacità degli elaboratori di testi, una fra tutte quella di segnalare e di correggere in automatico gli errori di ortografia. Essere in grado di

Figura **12**
Androide: uno degli obiettivi dell'Intelligenza Artificiale era anche quello della realizzazione di androidi, cioè robot con fattezze più o meno umane, in grado di svolgere compiti complessi, come e meglio di quanto non faccia l'uomo.



segnalare all'utente una parola contenente errori ortografici, come in questo caso:

```
quel romo del lago di Como
```

sembrava indicare una capacità quasi umana dei software di *word processing*.

Chiunque avesse buone conoscenze di programmazione comprendeva facilmente che lo strafalcione era stato rilevato grazie all'opera di un **analizzatore** - sostanzialmente un automa a stati finiti - che estraeva ogni parola della frase e la confrontava con quelle contenute in un vocabolario della lingua di riferimento, il tutto a velocità elevatissima.

Dato che nel dizionario non esisteva la parola `romo`, essa veniva segnalata come errata.

Al contrario, una frase come questa:

```
melo voglio regalare
```

non avrebbe dato luogo ad alcuna segnalazione da parte dell'analizzatore, mentre è evidente che contiene un errore: certamente l'autore intendeva scrivere

```
me lo voglio regalare
```

La differenza di valutazione della correttezza della frase deriva dal fatto che il termine `melo` è presente nel dizionario di riferimento del *word processor*, anche se indica la pianta che produce le mele. Tuttavia, essendo presente nel dizionario, "melo" è considerato un termine consentito e quindi non viene segnalato (anche se, non avendo alcuna attinenza con il contenuto, il senso della frase ne viene gravemente compromesso).

Comprendere • Come già abbiamo più volte ribadito in questo testo, un elaboratore non è in grado di *comprendere* il senso di un testo e non può fare altro che eseguire un codice, cioè applicare delle regole che sono state inserite a livello di programma sul contesto su cui opera. È un semplice esecutore e non possiede quella capacità che si intende con il termine "*comprensione del testo*".

Modelli di memoria • Per molto tempo si pensò alla mente umana come a una sorta di computer, e al cervello come a un processore in grado di gestire dati e informazioni e di registrarli nella nostra memoria sotto forma di *ricordi*.

In realtà, il funzionamento della memoria umana, come probabilmente quello della memoria degli animali superiori, è straordinariamente più complesso di quello degli elaboratori elettronici.

Cibernetica • Negli anni '60 del XX secolo grandi speranze furono riposte in una disciplina, la **Cibernetica**, che si occupava, tra le altre cose, di questo tipo di problematiche. In Italia, uno degli esponenti di spicco di questa materia fu Silvio Ceccato, al quale si deve il seguente bellissimo esempio che mostra quanto sia limitata la nostra conoscenza sul funzionamento della memoria.

Figura **13**
Silvio Ceccato
(1914 - 1997).



Supponiamo di voler trasmettere a un'altra persona una informazione su un libro noto, ad esempio *I Promessi Sposi* di Alessandro Manzoni.

Ci sono molti modi per farlo: potremmo scrivere un trattato in quattro volumi in cui si analizzano criticamente gli aspetti letterari e storici dell'opera: finiremmo forse per generare un'opera più ampia di quella di partenza.

In alternativa, potremmo riassumere in un paio di pagine, diciamo in 3.000 battute, le vicende narrate nel romanzo.

Se invece volessimo ridurre l'informazione ai minimi termini, potremmo sintetizzare l'opera con un'unica proposizione, ad esempio: "Storia di un amore contrastato tra due popolani nella Milano del Seicento, ai tempi della dominazione spagnola".

Ancora più sintetici: "Il capolavoro di Alessandro Manzoni".

In tutti i casi, stiamo parlando dello stesso argomento - e il nostro interlocutore lo capirà perfettamente - ma la nostra mente ha elaborato le conoscenze sull'opera manzoniana e le ha concretizzate in quattro modi completamente diversi. Come implementare la stessa capacità in un computer?

Al contrario, se ci si domandasse quante volte la lettera "f" compare nel testo manzoniano, trovare la risposta costituirebbe una fatica improba per un lettore umano, mentre anche il PC più modesto sarebbe in grado di rispondere velocemente e senza fatica.

È intuitivo, però, che il conteggio delle lettere che compongono un testo è un'operazione neppure paragonabile con il compito di descrivere l'opera di Manzoni: c'è un'enorme differenza nelle prestazioni richieste in termini di "comprensione" e quindi di intelligenza "autentica".

Non sappiamo come la nostra mente riesca ad organizzare le informazioni memorizzate e a recuperarle poi in modi tanto diversi eppure sempre significativi. Quindi non siamo neppure in grado di costruire un sistema artificiale in grado di eseguire lo stesso tipo di estrazione di informazioni da un contesto.

Ciò di cui siamo sicuri è che la mente non si limita ad accumulare informazioni come fossero tanti raccoglitori posti sugli scaffali di un archivio, ma sa usare a livelli diversi il loro contenuto di informazione ed è capace di riorganizzarne l'estrazione con metodi e modalità anche molto diversi tra loro.

5 Il Test di Turing

Alan Turing provò a dare una risposta al quesito che abbiamo visto al paragrafo precedente:

"***Può una macchina pensare?***"

Date le oggettive difficoltà nel definire in termini rigorosi cosa si possa intendere per "*macchina pensante*" o su come definire l'azione del "*pensare*", Turing riformulò il quesito in termini equivalenti, ma meno astratti:

"***Possono le macchine fare ciò che noi umani, intesi come esseri pensanti, siamo capaci di fare?***"

Con questo, lo scienziato britannico poneva a se stesso ed alla comunità scientifica una sfida: era possibile creare, anche se solo per ambiti limitati, una macchina tale per cui fosse impossibile, per chi interagiva con essa, decidere se aveva di fronte un essere umano o un computer?

Il problema era già stato posto due secoli prima dal filosofo illuminista Denis Diderot, il quale aveva affermato che, se avesse mai incontrato un pappagallo capace di rispondere a qualsiasi sua domanda, non avrebbe esitato a definirlo un essere intelligente.

Un gioco di società • Per cercare la risposta al quesito, Turing propose la questione nei termini seguenti: immaginiamo di giocare a un gioco di società, che possiamo chiamare "*Imitation game*" (gioco dell'imitazione). In questo gioco, A (un uomo) e B (una donna), invitati a una festa, si nascondono in due stanze diverse, senza che gli altri ospiti sappiano in quale stanza si trovi l'uno e in quale l'altra.

Gli invitati devono scoprire in quale stanza c'è l'uomo e in quale c'è la donna. A questo scopo possono fare domande ai due scrivendole su foglietti che fanno passare sotto la porta delle due stanze. Dal canto loro, A e B sono obbligati a rispondere: le risposte - scritte a macchina per maggior sicurezza - sono fatte passare sotto la porta. Analizzando le risposte, gli ospiti cercano di indovinare il sesso di chi si nasconde dietro ciascuna porta. Per complicare il gioco, A deve fingere di essere una donna e B deve fingere di essere un uomo.

Una volta descritte le regole del gioco, Turing pose la domanda chiave:

"*Cosa succederebbe se una macchina prendesse il posto di una delle due persone nascoste? Gli ospiti incontrerebbero più difficoltà nel determinarne il sesso per il fatto che le risposte sono generate da una macchina?*"

Il test ha senso? • Turing - e altri scienziati dopo di lui - raffinarono la formulazione del test in molti modi. Incontrarono anche numerose critiche: l'osservazione più ricorrente affermava che il *test non doveva scoprire se una macchina potesse essere intelligente quanto un umano*, bensì *doveva verificare se una macchina fosse in grado di emulare il comportamento umano*.

La differenza può sembrare sottile ma è in realtà profonda.

6 La sfida di Turing

Come il gioco degli scacchi affascina milioni di giocatori, pur non avendo componenti legate al caso o alla sorte, ed essendo governato da poche regole che si imparano, almeno formalmente, in breve tempo, così la sfida lanciata dal quesito di Turing, che peraltro non è ancora conclusa, ha incontrato, nel corso degli anni numerosi scienziati ed appassionati che hanno cercato di trovare una risposta valida.

Un software "psicoterapeuta" • Uno dei risultati più interessanti emersi nel corso della sfida è quello rappresentato da ELIZA, un programma scritto intorno alla metà degli anni '60 da Joseph Weizenbaum, e divenuto subito un caso.



Figura **14**
ELIZA a terminale negli anni '60.



Figura **15**
Il contesto di ELIZA.

Weizenbaum programmò ELIZA, mettendolo in grado di interagire con una persona (il "paziente") recitando la parte di uno psicoterapeuta.

Lo scenario ipotizzato da Weizenbaum è quello dello studio di uno psicoterapeuta, in cui il paziente, anziché disteso sul consueto lettino, siede davanti al terminale di un computer con il quale è in grado di interagire con un medico che si trova, invisibile, in un'altra stanza. Usando la tastiera, il paziente risponde alle domande che lo specialista gli pone. Il paziente non sa che il suo interlocutore è una macchina.

In questo contesto, il quesito di Turing si ripropone in questo modo:

"***È possibile scrivere un software in grado di rispondere alle domande del paziente***

(umano) in modo tale che egli non si accorga che sta interagendo con una macchina, ma sia convinto di avere a che fare con un medico in carne ed ossa?"

Esaminiamo un tipico scambio di battute tra ELIZA (lo psicoterapeuta) e il paziente. ELIZA comunica con il mondo esterno in inglese, cosa che ne ha reso più semplice la programmazione, data la maggiore essenzialità della grammatica e della sintassi della lingua inglese rispetto, ad esempio, a quelle neolatine.

Per comodità, sotto ogni frase abbiamo messo la traduzione in italiano.

ELIZA: Please tell me what's been bothering you.
Per cortesia, mi dica cosa c'è che non va.

Paziente: I feel terribly depressed.
Mi sento tremendamente depresso.

ELIZA: Do you enjoy feeling depressed?
Ti piace sentirti depresso?

Paziente: No.
No.

ELIZA: You are a bit negative.
Sei un po' negativo.

Paziente: What should I do?
Cosa dovrei fare?

ELIZA: That's quite interesting.
Ecco un punto interessante.

Paziente: What do you mean?
Cosa vuoi dire?

ELIZA: What are your feelings now?
Quali sono in questo momento le tue sensazioni?
....

Il comportamento di ELIZA è concettualmente semplice. Il dialogo comincia con una frase fatta piuttosto banale, che viene estratta a caso da un elenco di frasi di apertura, del tipo:

Buongiorno, è arrivato qui da solo?
Oggi mi piacerebbe che lei mi parlasse della sua infanzia.
Buongiorno, si accomodi e mi parli del suo problema.
La ascolto, mi dica cosa c'è che non va.
Ben arrivato, spero di poterla aiutare. Da dove iniziamo?

Un riconoscitore lessicale (quindi un automa a stati finiti) analizza ogni risposta digitata dal paziente, estrae e classifica le componenti in termini di sostantivi, verbi, avverbi, parole chiave, prende una di queste parole e, a partire da questa, costruisce una frase standard e la rilancia al paziente. Cerchiamo di scoprire il gioco di ELIZA.

Se analizziamo quanto dice ELIZA nell'ultima frase del dialogo, ci accorgiamo che dopo il verbo *feel* (sentirsi), ha estratto la parola *depressed* (depresso).

Feel è un verbo generico ed appartiene alla categoria dei verbi che descrivono una sensazione: il contenuto informativo della comunicazione del paziente risiede quindi sul fatto che

egli "si senta" in un determinato modo, che viene esplicitato dal termine che si trova subito dopo *feel*.

Nella frase, dopo *feel* compare *depressed*: ecco il termine chiave su cui ELIZA può costruire la domanda successiva, che è: *ti piace sentirti depresso?*

Se il paziente avesse scritto *solo* anziché *depresso*, probabilmente ELIZA avrebbe risposto: *ti piace sentirti solo?*

Quando le frasi sono troppo brevi, come nel caso della laconica risposta *no*, ELIZA non può usare la stessa metodologia: accede allora a un repertorio di frasi precostruite, da cui estrae una delle frasi che rimandano all'idea di negatività.

Il linguaggio di ELIZA, quindi, è ben costruito e, grazie ad una serie di semplici regole, genera frasi in *linguaggio naturale*, che sembrano del tutto pertinenti con la conversazione in atto.

In caso di una risposta sconcertante, ELIZA non sa come comportarsi ma è opportunamente programmata per cambiare argomento.

Molto umano, no?

B3-02
ELIZA (in Java)

Proponiamo online il link a una versione di ELIZA scritta in Java, accessibile online, che permette di interagire con il terapeuta sintetico. Vi invitiamo a provarci e ad analizzarne il comportamento: ne vale la pena.

Il trucco • In realtà, il cosiddetto terapeuta *sembra* comunicare con il paziente, ma non si tratta di un passaggio di informazioni bidirezionale.

È solo un osservatore delle affermazioni del paziente, che utilizza per riproporre quesiti o suggerire che il paziente dia ulteriori informazioni, senza mai fornire della vera informazione aggiunta.

È un vecchio trucco utilizzato anche da quanti, astrologhi, cartomanti e simili, cercano di far parlare chi sta ponendo loro un quesito sul proprio futuro o sulla propria vita, estraendo le parole chiave e costruendo mezze risposte a partire da frasi standard, in qualche caso volutamente arcane, in cui vengono reinserite le parole chiave che l'interrogante ha inconsapevolmente usato, spingendolo a dire di più.

Parlare o comunicare • L'analisi di un software tipo ELIZA ci permette anche di riconsiderare, con attenzione, molti dei dialoghi costruiti a tavolino, con una tecnica precisa, volta a far durare la trasmissione un numero quasi infinito di puntate, le battute dei dialoghi dei personaggi delle *telenovelas* o di tanti romanzi di nessun valore letterario che infestano le librerie dei luoghi di vacanza.

Eliza e il Test di Turing • Possiamo quindi dire che ELIZA è la dimostrazione che esiste almeno un programma in grado di superare il Test di Turing, cioè di ingannare un umano e fargli credere di interloquire con una persona mentre sta interagendo con una macchina?

La risposta è negativa: ponendo abilmente le domande ed analizzando la sequenza delle risposte, anche se con un certo sforzo, si giunge a comprendere che ELIZA è un sistema artificiale, per cui anche questo programma, pure importante sotto il profilo filosofico dell'analisi della comunicazione interpersonale, non supera il Test di Turing.

Concetti essenziali

- La formulazione di un problema non garantisce a priori l'esistenza della sua soluzione.
- Dato un codice generico P, non è garantito che esista un programma H che possa predire se il programma P, con determinati dati in ingresso, terminerà correttamente la propria esecuzione senza andare in loop.
- Esiste una classe di problemi, detti *indecidibili*, per i quali non esiste soluzione, indipendentemente dalla potenza di calcolo, dalla dimensione della memoria, dal numero di elaboratori e dal tempo impiegato per cercare la soluzione.
- Una Macchina di Turing è un automa a stati finiti.
- Una funzione è computabile se può essere risolta da una Macchina di Turing.
- Il tema fondante dell'Intelligenza Artificiale è: "Può una macchina pensare?"
- Con il Test di Turing si analizzano le capacità dei sistemi informatici di emulare il comportamento umano, almeno per alcune categorie di problemi ed in determinati ambiti.
- Allo stato attuale, non esiste ancora un programma che abbia superato positivamente il Test di Turing.

Test

1 **Dire se le seguenti affermazioni sono vere o false.**
L'affermazione "Dato un codice generico P, cioè un programma, è sempre possibile scrivere un programma H che analizzi P e ci dica se P, qualsiasi siano i dati in ingresso, terminerà la propria esecuzione correttamente o entrerà in loop?" è:
A vera V F
B falsa V F
C senza senso V F
D applicabile solo a codici scritti in linguaggi ad oggetti V F

2 **Dire se le seguenti affermazioni sono vere o false.**
I problemi per i quali non esiste risoluzione, indipendentemente dalla potenza di calcolo, dalla dimensione della memoria, dal numero di elaboratori e dal tempo che vengano impegnati per cercare tale soluzione sono detti:
A irrazionali V F
B impossibili V F
C indecidibili V F
D imbattibili V F

3 **Dire se le seguenti affermazioni sono vere o false.**
La Macchina di Turing è un dispositivo:
A virtuale V F
B variabile V F
C virtuoso V F
D virale V F

4 **Dire se le seguenti affermazioni sono vere o false.**
Nella Macchina di Turing, i dati in ingresso si immaginano sottoposti alla testa di lettura attraverso:
A un nastro suddiviso in celle quadrate; in ogni cella è presente un simbolo V F
B una tastiera e un mouse V F
C una chiavetta USB V F
D un accesso Web a uno specifico sito V F

5 **Dire se le seguenti affermazioni sono vere o false.**
Una Macchina di Turing è a tutti gli effetti:
A un metodo di test per gli algoritmi V F
B un linguaggio per l'iperconnettività V F
C un robot autosufficiente V F
D un automa a stati finiti V F

6 **Dire se le seguenti affermazioni sono vere o false.**
Il Test di Turing è espresso dalla seguente frase:
A "Possiamo insegnare agli uomini a fare qualsiasi tipo di calcolo più velocemente di quanto un computer possa fare?" V F
B "Possiamo progettare macchine capaci di fare qualcosa più velocemente di quanto noi uomini siano capaci di fare?" V F
C "Possono gli uomini fare ciò che i computer, intesi come esseri pensanti, sono capaci di fare?" V F
D "Possono le macchine fare ciò che noi umani, intesi come esseri pensanti, siamo capaci di fare?" V F

7 **Dire se le seguenti affermazioni sono vere o false.**
Attualmente, il Test di Turing:
A non è stato superato da alcun tipo di codice V F
B non ha più senso con gli attuali elaboratori V F
C è stato superato con l'introduzione dei tablet V F
D non viene più preso in considerazione in ambito informatico V F

Esercizi

1 Dividendo la classe in gruppi, e ogni gruppo in due sottogruppi, "macchina" ed "umani", si simuli verbalmente il comportamento di un software del tipo di quello di ELIZA e si verifichi come possano esistere strategie di interazione, da parte degli umani, per smascherare la "macchina" con cui si sta comunicando.

2 Riprendendo l'esercizio precedente, si provi a considerare la diversa situazione in cui ci si verrebbe a trovare se, anziché dover solo rispondere alle domande della macchina, anche l'umano potesse formulare a propria volta domande alla macchina.

3 Riprendendo l'esercizio di cui al punto 1, si provi a considerare le due diverse situazioni in cui ci si verrebbe a trovare imponendo alla macchina la regola di dire sempre la verità oppure di poter anche mentire, in risposta alle domande di un umano.

Sezione **C**

Fondamenti di calcolo numerico

1 Introduzione al calcolo numerico

Un elaboratore digitale, fondamentalmente, permette di risolvere problemi di natura numerica. Il capitolo introduce i concetti di soluzione simbolica e soluzione numerica, mostrando come l'introduzione dell'Informatica abbia dato nuovo impulso a studi risalenti al passato, aprendo nuovi orizzonti alle applicazioni del calcolo numerico e rendendolo un argomento molto più importante, nell'ambito attuale della matematica, di quanto non fosse sino alla prima metà del Novecento.

Viene presentato un linguaggio di programmazione di pubblico dominio, denominato R, che verrà poi usato nei prossimi capitoli per affrontare l'impostazione e la soluzione di problemi matematici per via numerica.

1 Problemi e soluzioni

Un po' di storia • Fin dalla notte dei tempi l'uomo ha cercato di affrontare le sfide che la realtà quotidiana gli imponeva, cercando di elaborare comportamenti e di sviluppare strategie che gli permettessero, in primo luogo, di preservare se stesso, la sua tribù e il suo territorio e inoltre di soddisfare i bisogni primari, quali la sopravvivenza alimentare, la riproduzione e la tutela della prole. Se il problema del primitivo era la necessità di cibo, la soluzione era raccoglierlo o catturarlo per potersene cibare, e a questo scopo non era neppure necessario saper contare sulle dita di una mano. Con il progredire della civiltà, all'aumentare della complessità delle relazioni sociali, i problemi divennero sempre più legati alle relazioni di convivenza e ai rapporti interpersonali. Si sviluppò il concetto di proprietà, e il calcolo numerico divenne indispensabile: ad esempio, l'istituto della proprietà terriera fu alla base della geometria (dal greco γεωμετρία, in cui γεω, gèo, significa terra, e μετρία, metrìa, misurare). Già ai tempi dell'antico Egitto, quando il Nilo periodicamente copriva di limo i terreni cancellando i confini dei singoli possedimenti, furono messi a punto procedimenti che permettevano di ricostruire le aree di pertinenza dei diversi proprietari: i procedimenti matematici utilizzati dagli Egizi sono la base della moderna topografia.

Più in generale, nel mondo antico, i problemi di natura pratica - che miravano ad ottenere un risultato numerico - richiedevano procedure di calcolo che portassero a un risultato in un tempo ragionevole sfruttando le limitate risorse di calcolo allora disponibili. Ad esempio, la suddivisione in proporzioni predeterminate dei frutti di un raccolto o delle prede catturate in una battuta di caccia presupponevano la padronanza del concetto di numero e una capacità - pur minima - di computazione: almeno quella di saper determinare l'ammontare del totale e di eseguire una divisione.

La scuola greca

La nascita della scienza matematica quale oggi la intendiamo è ascrivibile alla scuola greca, che non si limitò a trovare la soluzione contingente al problema del momento, ma passò ad un livello superiore di astrazione, sforzandosi di generalizzare i singoli problemi, e quindi di trovare procedure di calcolo svincolate dal particolare contesto del caso in esame.

I grandi nomi della Matematica



Pitagora di Samo (570 a.C. ca. - 495 a.C. ca.) fu filofoso e matematico, fondatore a Crotone di una scuola in cui la conoscenza e lo studio della matematica erano elementi basilari, dato che il numero era considerato fondamento della realtà.

Noto per il Teorema che porta il suo nome, studiò i numeri e le loro proprietà. Si interessò ai rapporti armonici e quindi alla musica. La sequenza dei primi quattro interi, 1, 2, 3 e 4, nota con il nome di Tetraktys, era nella sua visione tanto importante che egli stesso ed i suoi allievi, quando giuravano, dichiaravano di farlo nel nome della sacra Tetraktys.

Figura **1**
Pitagora.

Isaac Newton (1642 - 1727) fu il più grande scienziato britannico. Fu insigne matematico, filosofo e fisico, oltre che uno straordinario tecnologo, avendo inventato il primo telescopio a riflessione. In campo matematico, fu tra i padri dell'analisi infinitesimale, insieme a Leibniz, mentre come fisico pubblicò le leggi del moto e della gravitazione universale, fornendo un modello che rimase insuperato fino alla relatività di Einstein.

Ottenne risultati importanti in tutti gli ambiti in cui operò: fu grande teorico ma anche pragmatico inventore di metodi pratici, quali quelli per il calcolo numerico degli integrali definiti.



Figura **2**
Isaac Newton.



Carl Friedrich Gauss (1777 - 1855) fu uno scienziato tedesco che diede straordinari contributi sia in campo matematico sia in quello fisico, astronomico e delle scienze della terra. I suoi contemporanei lo definirono "Principe dei matematici", per la genialità delle sue dimostrazioni e per le molteplici scoperte che egli fece in questo campo. In ambito aritmetico, si interessò anche al calcolo numerico, al quale dedicò alcune opere.

Figura **3**
Carl Friedrich Gauss.

2 Natura delle soluzioni

Definizione di soluzione • Già nel mondo antico si andava delineando il concetto generale di *soluzione*: dato un problema di natura matematica - o esprimibile attraverso un modello che ne permetta un approccio di tipo computazionale - possono essere considerate soluzioni di esso tutti quei valori che soddisfano le relazioni espresse dalla formulazione del problema, tenendo conto della eventuale presenza di condizioni particolari, dette condizioni al contorno. Questo termine esprime un concetto di natura generale, non limitato ai soli problemi di natura matematica: per fare un esempio, nell'analisi dei dati relativi agli abitanti residenti in un dato territorio, sono condizioni al contorno innanzi tutto la certezza che i soggetti in esame siano viventi e poi che siano iscritti all'anagrafe di uno dei comuni sotto esame.

Esistenza e numero delle soluzioni • Un problema descrivibile in termini matematici non sempre - e non necessariamente - ha una soluzione. Talvolta, poi, le soluzioni possono essere più di una o, come vedremo in seguito, addirittura infinite.

Soluzioni simboliche e soluzioni numeriche • Un semplice esempio ci verrà in aiuto. Consideriamo l'espressione generale delle equazioni lineari di secondo grado:

$$ax^2 + bx + c = 0 \; (\text{con } a \neq 0)$$

Le soluzioni, come è noto, sono espresse dai valori x_1 ed x_2 che soddisfano le seguenti condizioni:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Le due relazioni che esprimono le soluzioni dell'equazione hanno validità generale: qualunque siano a, b, e c, sostituendo ai loro simboli i corrispondenti valori dell'equazione potremo calcolare i valori di x_1 e di x_2 (ricordiamo che le soluzioni sono reali se - e solo se - il discriminante è maggiore o uguale a zero).

Se per esempio fosse data l'equazione

$$14x^2 - 10x + 1 = 0$$

essendo $a = 14$, $b = -10$ e $c = 1$
la soluzione x_1 varrebbe:

$$x_1 = \frac{10 + \sqrt{100 - 56}}{28} \qquad \text{e cioè} \qquad x_1 = \frac{10 + 2\sqrt{11}}{28} = \frac{5 + \sqrt{11}}{14}$$

e la soluzione x_2 varrebbe:

$$x_2 = \frac{10 - \sqrt{100 - 56}}{28} \qquad \text{e cioè} \qquad x_2 = \frac{10 - 2\sqrt{11}}{28} = \frac{5 - \sqrt{11}}{14}$$

Vediamo, quindi, che la rappresentazione di x_1 e x_2 in funzione dei parametri a, b e c risolve il problema a livello simbolico, quindi generale, mentre la sostituzione dei parametri a, b e c con i valori specifici dell'equazione data permette di trovare i valori numerici che la risolvono.

Calcolo formale • La ricerca di procedure di risoluzione di funzioni a livello generale, cioè indipendentemente dai valori numerici dei parametri che le esprimono, prende il nome di **calcolo simbolico** o **calcolo formale**.

Dal particolare al generale • Poter disporre di una procedura generale di risoluzione di una classe di problemi rappresenta, come è facile intuire, un punto di forza: verificate le condizioni di applicabilità del modello, si tratta di sostituire ai parametri generali quelli dello specifico caso e di procedere alle operazioni necessarie per determinare il risultato. A volte, i calcoli necessari sono di livello elementare, ma in molte situazioni l'espressione della soluzione può essere assai complessa.

Se la soluzione di un dato problema fosse data da un'espressione quale la seguente:

$$y = \frac{ax^7 + \frac{b}{c}x^6 + \frac{d}{1000\pi}x^4 + \sqrt{ex^8 - \log\left(x\sqrt{\frac{fx^{13}}{gx^7 - 3}}\right)}}{\log\left(\sqrt[5]{x^4 + 87x^{11}}\right)}$$

per $x = 5$, in cui avessimo:

$$a = 20$$
$$b = 1$$
$$c = 8$$
$$d = 14.3$$
$$e = -2$$
$$f = 30$$
$$g = 101$$

dato un certo valore di x, la ricerca del valore di y con carta e penna rappresenterebbe una fatica non indifferente.

Nel passato • Anticamente, la complessa numerazione romana (additiva) rendeva particolarmente disagevole eseguire le quattro operazioni. Per questo, dove possibile, invalse l'uso di scegliere un approccio di tipo geometrico, evitando complessità computazionali. Quando poi questo non era possibile, veniva fatto largo uso di tavole numeriche.

In alcuni casi, la complessità dei calcoli necessari per giungere al risultato era tale da scoraggiare i matematici e da far optare loro per una scelta diversa (di cui parleremo nei capitoli seguenti): si rinunciava a una soluzione matematicamente ineccepibile in favore di un risultato che si potesse ottenere con un minor lavoro di computazione, purché l'approssimazione del risultato rimanesse entro limiti accettabili.

Calcolo numerico • Quanto detto ci porta a definire il calcolo numerico come quella branca della matematica che si occupa dello studio delle procedure di risoluzione dei problemi per via computazionale e della valutazione dell'approssimazione dei risultati.

Quest'ultimo aspetto non è solo un esercizio interessante dal punto di vista teorico: a tutt'oggi un gran numero di problemi matematici esprimibili formalmente non hanno trovato soluzioni di carattere generale, ed è perciò necessario accettare un certo livello di approssimazione del risultato: ricordiamo la ricerca delle soluzioni di equazioni polinomiali di grado superiore al quinto e degli autovalori di matrici di grandi dimensioni.

Sistemi di equazioni lineari

Consideriamo un semplice sistema di equazioni di due equazioni lineari in due variabili, ad esempio:

$$2x + 3y = 6$$
$$4x + 9y = 15$$

La ricerca delle soluzioni di un tale sistema avviene normalmente attraverso un semplice calcolo per sostituzione:

$$x = 3 - \frac{3}{2}y$$

$$4\left(3 - \frac{3}{2}y\right) + 9y = 15$$

da cui si ricava

$$y = 1$$
$$x = 3/2$$

Nei sistemi di equazioni, al crescere del numero delle equazioni e delle variabili il metodo di risoluzione per sostituzione diventa inapplicabile: per la ricerca delle soluzioni si utilizzano comunemente metodi basati sulla rappresentazione matriciale del sistema in esame.

Operando sulla matrice di rappresentazione è possibile, sotto date condizioni, ottenere i risultati richiesti con l'aiuto di un elaboratore.

A titolo di esempio, consideriamo il seguente sistema di 3 equazioni in 3 incognite:

$$\begin{aligned} x + 3y - 2z &= 5 \\ 3x + 5y + 6z &= 7 \\ 2x + 4y + 3z &= 8 \end{aligned}$$

Se le equazioni sono scritte mantenendo le variabili sempre nello stesso ordine, come in questo caso, esso può essere rappresentato dalle matrici

$$\begin{bmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{bmatrix} \text{ e } \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$$

Esiste un metodo generale che permette la risoluzione di questo tipo di sistemi. La sua formalizzazione richiede però l'introduzione di concetti diversi, quali quello di ***matrice inversa***, di ***determinante*** e di ***autovalore***, tipici di quella branca della matematica che va sotto il nome di ***Algebra Lineare***.

Il numero di elaborazioni sugli elementi della matrice di rappresentazione per il raggiungimento del risultato cresce al crescere dell'ordine del problema: già sistemi di quarto ordine richiedono un numero di operazioni assai elevato.

Per molti problemi in cui i modelli di rappresentazione possono essere ben approssimati da sistemi di equazioni lineari in n variabili, il calcolo numerico diviene, di fatto, l'unica strada possibile per la ricerca delle soluzioni.

3 Computer e calcolo numerico

Rinascita del calcolo numerico • La disponibilità di strumenti informatici, quali i computer, per loro natura votati alla trattazione numerica dei dati, ha riportato alla ribalta, nel corso del XX secolo, l'interesse per la soluzione numerica dei problemi: per un matematico del XIX secolo, il calcolo a mano poteva richiedere un numero di passaggi la cui durata complessiva sarebbe stata inaccettabile - per non parlare del rischio di possibili errori. Al contrario, un computer può eseguire in frazioni di secondo milioni o miliardi di calcoli senza distrarsi, e questo rende più che accettabili i metodi numerici.

Applicazioni • Le applicazioni del calcolo numerico sono numerosissime e, in alcuni ambiti, insostituibili: ne sono esempi la ricostruzione per via digitale di immagini diagnostiche, come avviene in campo medico per la TAC (Tomografia Assiale Computerizzata) e per la Risonanza Magnetica, l'elaborazione delle immagini da satellite, i calcoli strutturali per l'ingegneria civile, e innumerevoli applicazioni in fisica, in chimica, in biologia, in statistica: le discipline scientifiche sono oggi più che mai sostenute, in termini computazionali, dall'uso del calcolo numerico.

Esempi di utilizzo • Classificati per tipologia, i principali problemi che possono essere risolti vantaggiosamente col calcolo numerico sono i seguenti.

- *Equazioni lineari*: la risoluzione generale di un sistema di *n* equazioni lineari in *n* incognite
- *Programmazione lineare*: la minimizzazione di una funzione lineare soggetta a un certo numero di vincoli
- *Ottimizzazione*: la minimizzazione di una funzione di più variabili
- *Calcolo differenziale*: la ricerca delle soluzioni numeriche di sistemi di equazioni differenziali alle derivate parziali

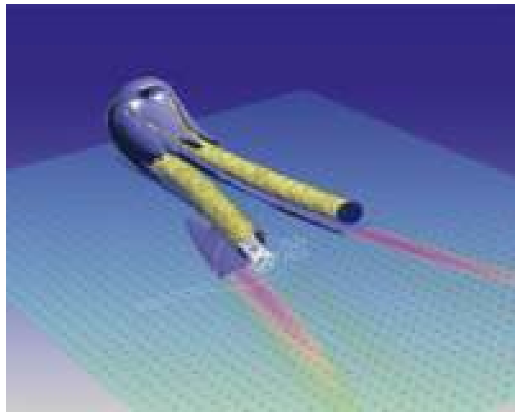


Figura **4**
Calcolo di moti vorticosi: attraverso una simulazione di tipo grafico vengono visualizzati i moti dei fluidi di un dispositivo a reazione in particolari condizioni di funzionamento.

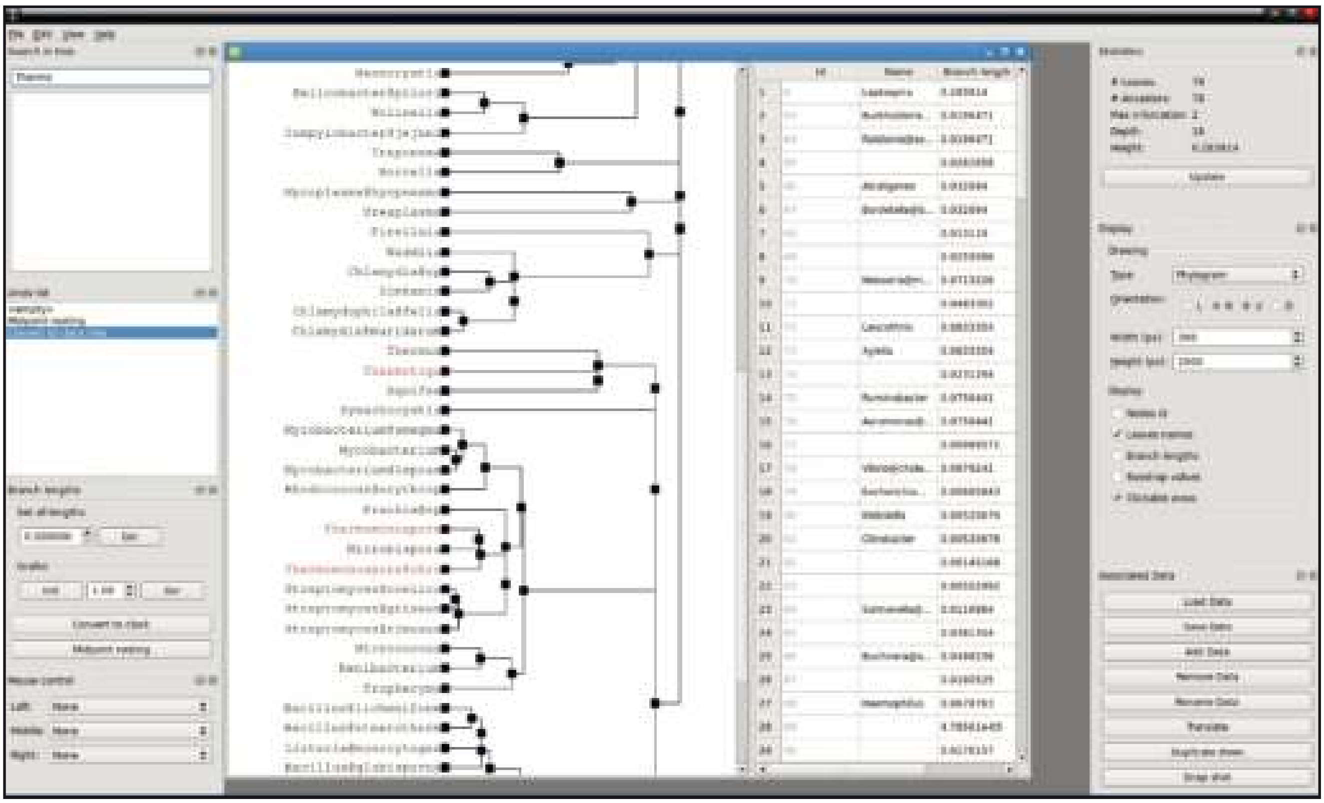


Figura **5**
La genetica necessita in moltissimi ambiti di potenza di elaborazione per estrarre informazione da enormi moli di dati

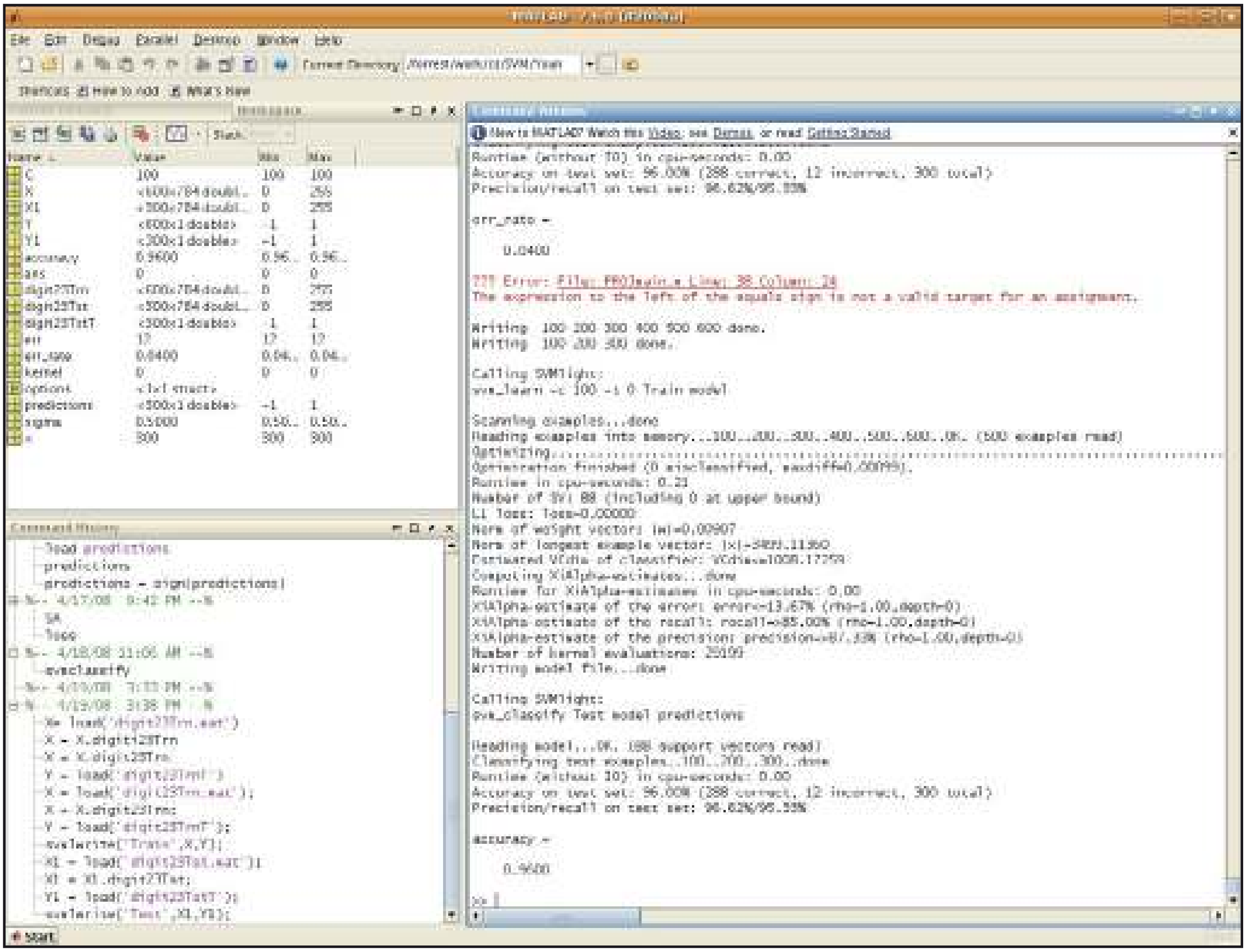
Scrivere software, usare software • Chiunque abbia un minimo di dimestichezza con un linguaggio di programmazione non avrà difficoltà ad impostare la relazione espressa dalla soluzione dell'esempio riportato nel box "Sistemi di equazioni lineari": il codice della sequenza del calcolo può essere impostato, ad esempio, in pseudo-C, o in un qualsiasi altro linguaggio idoneo che l'utente conosca.

L'ampiezza crescente della comunità scientifica, che necessita di eseguire elaborazioni di natura matematica anche complessa, ha portato alla nascita di piattaforme di calcolo dotate di potenti ed efficaci metodi di interfacciamento con l'utente. Attraverso essi è possibile impostare espressioni e ottenere risultati anche per problemi complessi senza essere specialisti di programmazione in linguaggi avanzati. Si intende che la soluzione deve essere determinabile per via matematica, anche eventualmente al prezzo di una certa approssimazione del risultato.

Questi programmi mettono a disposizione un gran numero di funzioni matematiche di tipologie diverse, e permettono all'utilizzatore di impostarle in modo interattivo o attraverso la realizzazione di script. Il risultato sarà visualizzato in forma numerica o, se richiesto, in forma di grafico.

MATLAB® e non solo • Al momento della stesura del presente testo, il linguaggio più utilizzato dai ricercatori e, in genere, da quanti necessitino di eseguire calcoli complessi, è MATLAB®, creato da MathWorks.

Figura **6**
Schermata Matlab.



MATLAB®

MATLAB® è un linguaggio di alto livello e un ambiente interattivo per il calcolo numerico, l'analisi, la visualizzazione dei dati e la programmazione. Sviluppato da MathWorks, azienda statunitense, offre potenti strumenti per il calcolo matriciale, il plottaggio di funzioni e di dati, la creazione di interfacce utente. Nato come strumento per il calcolo numerico, mette a disposizione, attraverso librerie opzionali, MuPAD, un motore di calcolo simbolico. Un'altra libreria, Simulink, permette la progettazione, attraverso modelli, di sistemi dinamici. La licenza d'uso è acquistabile a pagamento.

Alcune schermate del pacchetto MATLAB®

Per gli scopi di questo corso di Informatica è sufficiente utilizzare il pacchetto Open Source, denominato R, reperibile in rete all'indirizzo Web www.r-project.org

Tra le diverse alternative, disponibili anche su piattaforme diverse, troviamo, allo stato dell'arte, i pacchetti SciLab, Octave e FreeMat.

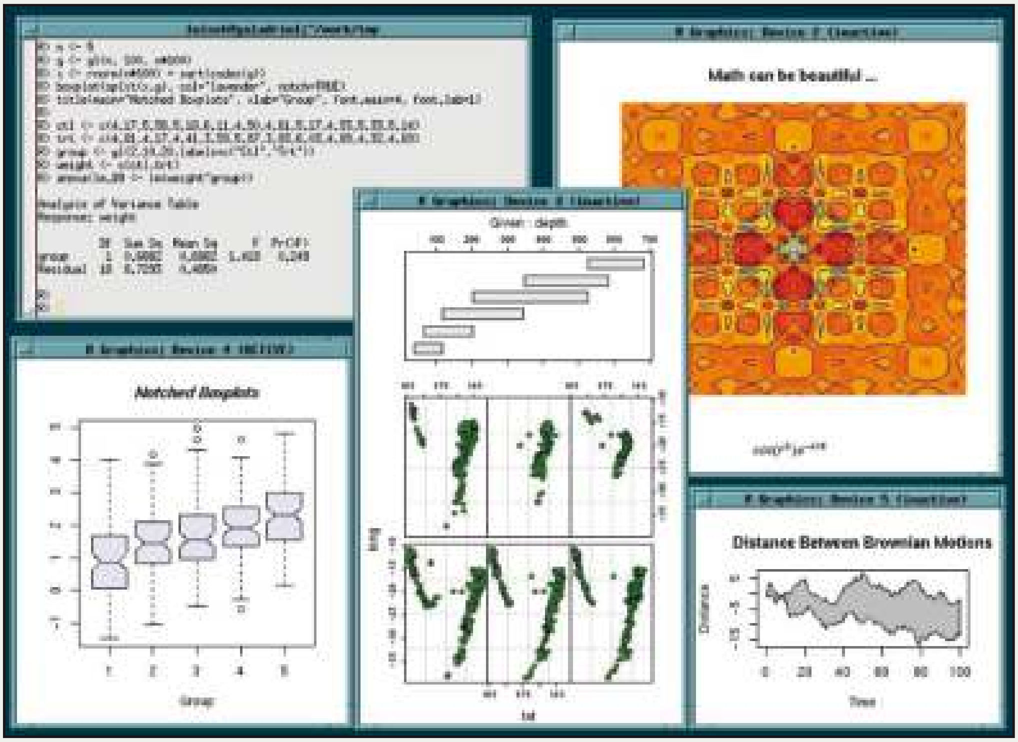


Figura **7**
Schermata R.

4 Primi passi con R

Nato inizialmente per fini statistici, il pacchetto R svolge comunque operazioni matematiche di ogni tipo: la sua sintassi è compatibile con quella di MATLAB® e, inoltre, offre il vantaggio di essere totalmente gratuito e più che valido per le applicazioni pratiche della maggior parte degli utenti.

Sintassi • Il linguaggio R è un linguaggio di programmazione orientato alle applicazioni scientifiche, con cui si può programmare in modo più diretto rispetto a quanto sarebbe necessario con C o con Java per ottenere lo stesso risultato. La disponibilità di un ricco portafoglio di funzioni matematiche e grafiche integrate lo rendono interessante in molteplici ambiti di applicazione. Alla sintassi del linguaggio R è dedicato un approfondimento disponibile online.

C1-01
La sintassi di R

Calcolo interattivo • R può essere utilizzato in modalità interattiva: se scriviamo, ad esempio:

```
> a = 5
> b = 3
> print (a+b)
```

otteniamo il seguente risultato:

```
> a = 5
> b = 3
> print (a+b)
[1] 8
> |
```

Creare e assegnare variabili • Se vogliamo tracciare il grafico di una funzione trigonometrica elementare, ad esempio:

$$y = \sin x$$

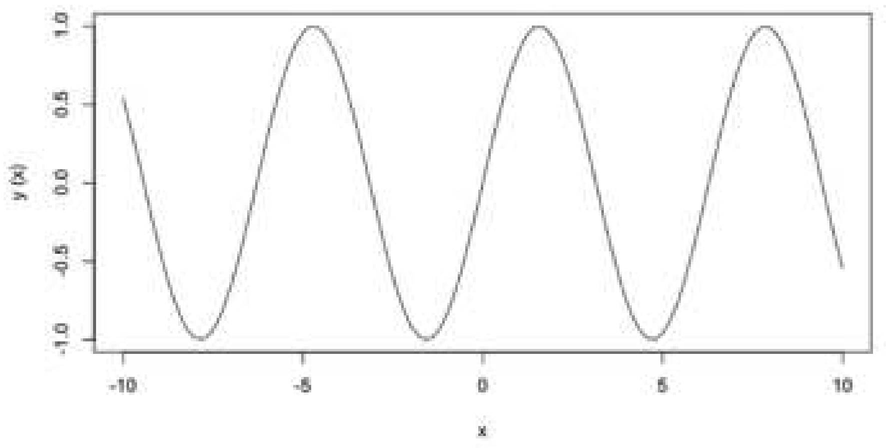
per prima cosa, dobbiamo definire le variabili di nostro interesse:

```
> y <- function(x)
+ y = sin(x)
> |
```

Poi, se vogliamo ottenere il grafico della funzione, ad esempio nell'intervallo `x` compreso tra `-10` e `+ 10`, utilizzeremo l'istruzione `plot` in questo modo:

```
> y <- function(x)
+ y = sin(x)
> plot(y,-10,+10)|
```

Il risultato a video sarà il seguente:



Un esercizio elementare • Per iniziare a familiarizzare con R, proviamo, in modalità interattiva, a tracciare i grafici, nell'intervallo `-100<x<100` di due curve analitiche: una parabola e una retta che la interseca.

L'equazione della parabola sia la seguente:

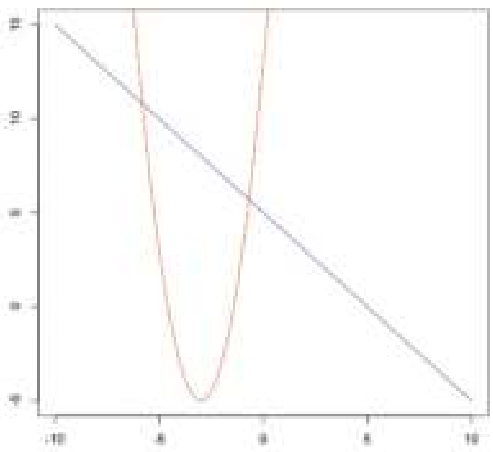
$$y = 2x^2 + 12x + 13$$

e quella della retta sia:

$$y = -x + 5$$

Impostando con R il problema nel seguente modo:

```
y2 <- function(x)
y2 <-(-x+5)
plot(y2,-10,+10, col="blue")
par(new=TRUE)
y1 <- function(x)
y1 <- (2*x^2+12*x+13)
plot(y1,-10,+10,col="red", add=TRUE)
```



otterremo il seguente risultato:

Notiamo come sono stati utilizzati i parametri di colore e l'uso di

```
par(add=TRUE)
```

per mantenere lo stesso spazio grafico di visualizzazione per le due funzioni, cioè per fare in modo che la seconda istruzione

```
plot()
```

non vada a tracciare la retta in una nuova finestra.

Per approfondimenti, si rimanda alla documentazione a disposizione sul sito www.r-project.org.

Concetti essenziali

- Il calcolo numerico e il calcolo simbolico sono argomenti trattati in ambito matematico.
- Attraverso il calcolo simbolico si cercano soluzioni di ordine generale, indipendenti dal valore delle variabili.
- Il calcolo simbolico è detto anche calcolo formale e le soluzioni che esso fornisce sono di carattere generale.
- Il calcolo numerico studia i metodi per risolvere i problemi per via numerica e per valutare l'approssimazione del risultato.
- Le soluzioni offerte dal calcolo numerico sono specifiche per ogni problema, poiché dipendono dai parametri delle equazioni risolutive e dai valori delle variabili.
- Il software R, di pubblico dominio, è compatibile a livello di codice con il noto e diffusissimo software MATLAB®.

Test

1 **Dire se le seguenti affermazioni sono vere o false.**
Qualsiasi problema esprimibile in termini matematici:
A ammette sempre una soluzione V F
B può essere risolubile sia simbolicamente che numericamente V F
C se ha una soluzione simbolica non può avere una soluzione numerica V F
D è sempre risolubile, almeno per via numerica V F

2 **Dire se le seguenti affermazioni sono vere o false.**
Il calcolo simbolico prende anche il nome di:
A calcolo formale V F
B calcolo analitico V F
C calcolo astratto V F
D calcolo intuitivo V F

3 **Dire se le seguenti affermazioni sono vere o false.**
L'uso dei computer:
A ha fatto passare in secondo piano il calcolo numerico V F
B ha fatto scoprire per la prima volta il calcolo numerico V F
C ha dimostrato l'inutilità del calcolo numerico V F
D ha fatto rinascere l'interesse per il calcolo numerico V F

4 **Dire se le seguenti affermazioni sono vere o false.**
Il linguaggio open source R usato in questo corso è:
A compatibile con MATLAB® a livello di codice V F
B utilizzabile solo con una licenza MATLAB® V F
C un sottoinsieme di MATLAB® sviluppato nelle università italiane V F
D disponibile solo per i tablet V F

5 **Dire se le seguenti affermazioni sono vere o false.**
R è un linguaggio di tipo:
A interattivo V F
B interoperativo V F
C interspecifico V F
D interdisciplinare V F

6 **Dire se le seguenti affermazioni sono vere o false.**
Il calcolo numerico:
A trova moltissime applicazioni in diversi campi, grazie all'uso estensivo degli elaboratori V F
B è una disciplina importante solo ed esclusivamente dal punto di vista teorico V F
C serve solo per applicazioni di tipo statistico V F
D è solo un'alternativa comoda alla ricerca della soluzione per via simbolica di ogni problema matematico V F

Esercizi

Attenzione: per la soluzione dei seguenti esercizi si faccia riferimento alla documentazione generale del linguaggio R, disponibile online sul sito del corso.

1 **Utilizzando il linguaggio R si scriva un breve programma che, ricevuti in ingresso 5 numeri interi positivi scelti a caso, li presenti a video in ordine crescente.**

2 **Riprendendo l'esercizio precedente, utilizzando il linguaggio R si modifichi il codice precedente in modo che riceva un numero qualsiasi (minore di 100) di numeri in ingresso, terminando l'input e procedendo all'elaborazione solo quando l'operatore digiti in input 0 o dopo l'inserimento del 100-simo numero.**

3 **Utilizzando il linguaggio R si scriva un programma che acquisisca da tastiera delle coppie di interi, ad esempio la sequenza seguente:**

1 1
2 8
3 27
4 64
5 125

li consideri come le coordinate x ed y di una funzione e mostri a video in forma grafica il tracciato della funzione corrispondente.

4 **Utilizzando il linguaggio R si scriva un programma in grado di tracciare a video la rappresentazione grafica della seguente funzione:**

$y = 4x^2 - 12x + 3$

nell'intervallo $-5 \leq x \leq 20$

5 **Utilizzando il linguaggio R, si scriva un programma in grado di tracciare a video in due diversi colori le rappresentazione grafiche delle seguenti funzioni:**

$y = 8x^3 + 10x^2 - 50x + 1$

e

$y = 54x^2 - 100x + 20$

nell'intervallo $-100 \leq x \leq +100$

6 **Utilizzando il linguaggio R, si scriva un programma che riceve in input 3 valori *a*, *b* e *c* e tracci a video la rappresentazione grafica della funzione:**

$y = ax^4 - 20x^3 + 12x^2 - 14x + 5$

nell'intervallo $-50 \leq x \leq +50$

2 Metodi diretti e iterativi

Con questo capitolo si entra nel vivo del calcolo numerico: dato un problema, andremo a creare un programma che ne fornisce la soluzione.

Il bimillenario problema del crivello di Eratostene viene utilizzato come esempio per mostrare come, usando un approccio di tipo numerico, si possa affrontare un quesito posto in termini generali e quali siano i limiti della soluzione ottenuta con questo approccio.

Con gli sviluppi in serie di Taylor e di Maclaurin vengono poi introdotti i metodi iterativi, che permettono di calcolare il valore che le funzioni matematiche assumono per in certi punti, nei quali il calcolo teorico incontra dei problemi. Viene inoltre presentato il concetto di serie convergente, che verrà approfondito successivamente.

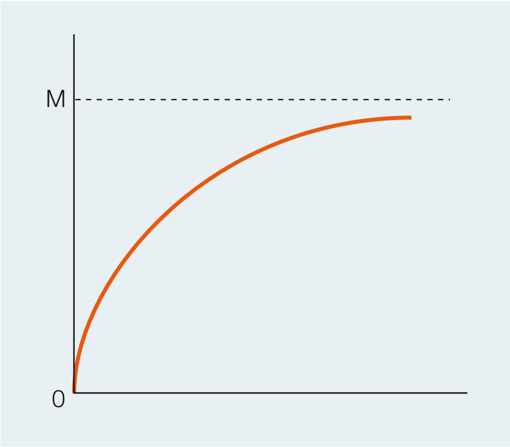
1 I metodi del calcolo numerico

Conoscere gli strumenti • Nell'utilizzo di R per la risoluzione dei problemi, noti i parametri, non resta che inserirli e iniziare a svolgere, o a far svolgere dal software, la sequenza delle operazioni programmate.

Le metodologie del calcolo numerico, tuttavia, richiedono uno studio più approfondito, sia per conoscerne le grandi potenzialità che per saperlo utilizzare nella piena consapevolezza dei suoi limiti.

Metodo diretto e metodo iterativo • Un metodo che prevede la risoluzione di un problema matematico con un algoritmo che fornisce il risultato in un numero finito di passi si definisce *metodo diretto*. Quando invece la risoluzione di un problema matematico avviene per approssimazioni successive e in un numero infinito di passi, si parla di *metodo iterativo*.

Non sempre l'esistenza di un metodo diretto lo farà preferire, ai fini pratici, a quello iterativo. Il metodo diretto, infatti, potrebbe essere troppo gravoso in termini di computo, oppure talmente oneroso in termini di tempo di calcolo da rendere inattuabile la sua applicazione. A sua volta, l'utilizzo di un metodo iterativo ha senso solo se al crescere del numero dei passi la funzione in esame è *convergente*.



Convergenza • In ambito matematico, viene detta **convergenza** la proprietà di una funzione - o di una **successione** - di raggiungere un valore limite finito al tendere della variabile all'infinito o a un valore determinato.

Figura **1**
Il grafico di una funzione convergente al valore M.

2 Il crivello di Eratostene

Un problema antico • Le proprietà dei cosiddetti *numeri primi* hanno incuriosito i matematici di ogni tempo. Ricordiamo che un numero si dice primo se - e solo se - ammette come divisori soltanto se stesso e l'unità.

Fin dai tempi dell'antica Grecia, uno dei più ricorrenti problemi relativo ai numeri primi può essere così formulato:

"*Dato un numero N prefissato a piacere, esiste un procedimento generale che permetta di determinare tutti i numeri primi il cui valore è inferiore a N?*"

Una soluzione geniale • Lo scienziato greco Eratostene studiò il problema e lo risolse con un metodo diretto, nel quale, se *N* è finito, il procedimento termina in un numero finito di passaggi inferiore a un valore determinabile a priori.

Il metodo si basava su un concetto fondamentale: applicare all'insieme dei numeri da 2 a *N* un criterio che permettesse di eliminare, per passaggi successivi, quelli sicuramente non primi: al termine del procedimento, i numeri rimanenti non potevano che essere quelli cercati. Questo procedimento richiamava alla mente l'azione di *setacciare*, con cui si separavano con un setaccio - detto anche *crivello* - i semi di cereali di diverse dimensioni. Analogamente, nel crivello di Eratostene, una lista dei numeri viene fatta passare attraverso un *crivello mentale*, in cui, ad ogni passaggio, si può aumentare la dimensione dei fori, fino a che nessuno dei numeri rimasti lo può più attraversare. La proprietà notevole di questo procedimento è il fatto che *il numero di passaggi è finito e predeterminabile*.

Eratostene di Cirene

Lo scienziato greco Eratostene, nativo di Cirene e vissuto nel III secolo a.C., fu poeta, matematico e astronomo. Inventore del termine "geografia", riuscì, con metodi geniali, a determinare la lunghezza della circonferenza della Terra e la distanza tra la Terra e il Sole. La storia della matematica lo ricorda come inventore del procedimento di calcolo detto "crivello di Eratostene".

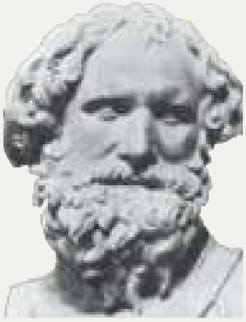


Figura **2**
Eratostene di Cirene.

Un esempio • Supponiamo di voler determinare tutti i numeri primi inferiori a 26. Il procedimento di Eratostene prevede che si ordinino in una tabella i numeri da 2 sino a 26.

Ecco come appare la tabella:

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26				

A questo punto si scartano tutti i multipli del primo numero riportato nella tabella (nell'esempio il numero 2), senza eliminarlo. In questo modo si eliminano tutti i numeri che hanno come divisore il numero 2, e che quindi non possono essere numeri primi.

Questo è il risultato (i numeri eliminati sono quelli su fondo scuro):

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26				

Ripetiamo l'operazione con il primo numero non scartato, cioè il 3 (che non va eliminato). Vengono scartati i numeri che hanno il 3 come divisore, e non possono essere quindi numeri primi. Ecco il risultato dopo il secondo passaggio:

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26				

Una volta compreso il procedimento, lo si applica finché nessun numero della tabella trova più un divisore. Il risultato finale - i numeri primi compresi tra 2 e 26 - è indicato dai numeri su fondo bianco della tabella:

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26				

Il numero di passaggi è prevedibile • È possibile dimostrare matematicamente che quando il *quadrato del numero primo* che viene usato come divisore è maggiore del numero *N*, non si ha più alcuna eliminazione. Si può quindi prevedere a priori il numero di passaggi necessario per completare il procedimento.

Nel nostro esempio, in cui $N = 26$, il primo dei numeri primi il cui quadrato è maggiore di 26 è il numero 7. Il procedimento perciò, termina dopo aver eliminato i numeri divisibili per 5, e cioè dopo tre passaggi.

Altri metodi • La soluzione di Eratostene non è la sola possibile, ma è particolarmente elegante e si mantiene efficiente per ricerche in cui *N* è inferiore a 10 000 000: per valori superiori, si possono applicare crivelli derivati, con criteri di eliminazione differenti, o metodi basati su altri criteri risolutivi.

3 Un crivello informatico

È evidente che, se *N* è un numero piccolo, il metodo di Eratostene è facilmente applicabile con carta e matita. All'aumentare di *N*, però, le cose si complicano: ecco che gli strumenti informatici divengono insostituibili.

Un programma in R • Qui di seguito potete trovare un semplice programma in R che applica il metodo ideato dal matematico greco. Nella figura 3 si può vedere il diagramma di flusso di questo "crivello informatico".

```
> ###### Crivello di Eratostene
> #
> # cerchiamo i numeri primi < 101
> N <- 101
```

```
> ##### VETTORE contiene i numeri da 1 a N
> VETTORE <- seq(1:N)
> #
> i <- 2
> while(i^2 < N){
+             for (j in (i +1):(N-1))
+             if((VETTORE[j] %% i) == 0)
+                  VETTORE[j] <- 0
+                  i <- i+1
+    }
> # stampiamo i risultati
> for (i in 2:(N-1)){
+    if (VETTORE[i] != 0){
+         print(VETTORE[i])}
+         }
```

Se proviamo a eseguire il programma, ad esempio, per $n = 594$, troveremo i seguenti risultati:

[1] 2	[1] 67	[1] 157	[1] 257	[1] 367	[1] 467
[1] 3	[1] 71	[1] 163	[1] 263	[1] 373	[1] 479
[1] 5	[1] 73	[1] 167	[1] 269	[1] 379	[1] 487
[1] 7	[1] 79	[1] 173	[1] 271	[1] 383	[1] 491
[1] 11	[1] 83	[1] 179	[1] 277	[1] 389	[1] 499
[1] 13	[1] 89	[1] 181	[1] 281	[1] 397	[1] 503
[1] 17	[1] 97	[1] 191	[1] 283	[1] 401	[1] 509
[1] 19	[1] 101	[1] 193	[1] 293	[1] 409	[1] 521
[1] 23	[1] 103	[1] 197	[1] 307	[1] 419	[1] 523
[1] 29	[1] 107	[1] 199	[1] 311	[1] 421	[1] 541
[1] 31	[1] 109	[1] 211	[1] 313	[1] 431	[1] 547
[1] 37	[1] 113	[1] 223	[1] 317	[1] 433	[1] 557
[1] 41	[1] 127	[1] 227	[1] 331	[1] 439	[1] 563
[1] 43	[1] 131	[1] 229	[1] 337	[1] 443	[1] 569
[1] 47	[1] 137	[1] 233	[1] 347	[1] 449	[1] 571
[1] 53	[1] 139	[1] 239	[1] 349	[1] 457	[1] 577
[1] 59	[1] 149	[1] 241	[1] 353	[1] 461	[1] 587
[1] 61	[1] 151	[1] 251	[1] 359	[1] 463	[1] 593

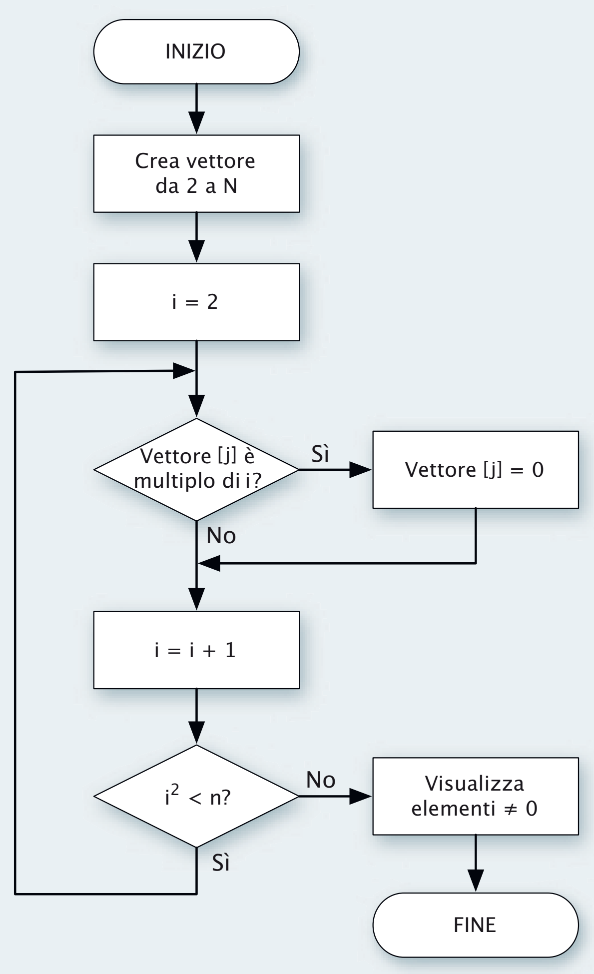


Figura **3**
Il diagramma di flusso del "crivello informatico".

4 Metodi iterativi

Esaminiamo ora i metodi iterativi. I due più noti prevedono la sostituzione della funzione in esame con una funzione diversa, meno gravosa dal punto di vista del calcolo, anche se penalizzata dal fatto che essa implica necessariamente una certa approssimazione del risultato.

Il metodo di Taylor • Uno dei metodi più noti per la ricerca delle soluzioni di una equazione attraverso una semplificazione della sua rappresentazione è quello noto sotto il nome di *metodo di Taylor*, dal nome del matematico inglese Brook Taylor (1685-1731).

Taylor dimostrò che una funzione $f(x)$ che sia infinitamente differenziabile nell'intorno di un punto può essere approssimata da una funzione polinomiale, in cui i termini da aggiungere al valore della funzione nel punto sono ottenuti a partire dalle derivate di ordine crescente della funzione di partenza.

Vediamo ora l'espressione generale della formula di Taylor.

Data una funzione $f(x)$ avente le caratteristiche sopra espresse, possiamo approssimarne il valore nel punto a come:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots$$

Utilizzando la notazione degli sviluppi in serie, possiamo scrivere, in modo più compatto, che:

$$f(x) = \sum_{n=1}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

In questa espressione la notazione $n!$ rappresenta il fattoriale di n, cioè il prodotto di tutti i numeri compresi tra 1 e quel numero stesso, e $f^{(n)}(a)$ è la derivata di ordine n-simo della funzione di partenza calcolata nel punto a.

Nel caso particolare in cui $a = 0$, la serie prende il nome di serie di Maclaurin.

Un esempio • Vediamo l'espressione della serie di Taylor che approssima la funzione

$$y = \sin(x)$$

nell'intorno del punto $x = 0$.

Siamo nel caso di applicazione della serie di Maclaurin: la sua espressione, limitata ai primi quattro termini, è la seguente:

$$f(x) = f(0) + \frac{f'(0)}{1!}(x) + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3$$

e ricordando che

$$f(x) = \sin(x) \qquad f'(x) = \cos(x) \qquad f''(x) = -\sin x \qquad f'''(x) = -\cos x$$

con le opportune sostituzioni si ottiene:

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

La precisione del risultato aumenta con il numero dei termini della serie, ma si può dimostrare che già per la funzione sopra riportata, per $-1 < x < +1$, l'errore commesso utilizzando per il calcolo la serie di Maclaurin è inferiore a

$$\frac{|x^9|}{9!}$$

quindi a 0,000003.

È importante notare come negli sviluppi in serie di questo tipo la funzione potrebbe non essere derivabile per valori di x isolati o per interi intervalli. In corrispondenza di essi le serie di Taylor o di Maclaurin non possono essere calcolate e perdono significato.

Alcuni sviluppi di Maclaurin • Di seguito riportiamo alcuni sviluppi di Maclaurin per funzioni di uso comune:

- Funzione esponenziale

e^x per qualsiasi x $$e^x = \sum_{n=1}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

- Funzioni del logaritmo naturale di x

$\log(1-x)$ per $|x|<1$ $$\log(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n}$$

$\log(1+x)$ per $|x|<1$ $$\log(1+x) = \sum_{n=0}^{\infty} (-1)^{n+1} \frac{x^n}{n}$$

- Funzioni trigonometriche di base:

$\sin x$ per qualsiasi x $$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$\cos x$ per qualsiasi x $$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

$\tan x$ per $|x| < \frac{\pi}{2}$ $$\tan x = \sum_{n=1}^{\infty} \frac{B_{2n}(-4)^n(1-4^n)}{(2n)!} x^{2n-1} = x + \frac{x^3}{3} + \frac{2x^5}{15} + \cdots$$

I termini B che compaiono nell'espressione del calcolo della tangente sono valori costanti, i cosiddetti numeri di Bernoulli, dal nome del matematico svizzero Jakob Bernoulli (1654-1705).

I primi otto numeri di Bernoulli sono i seguenti:

$$B_0 = 1 \qquad B_1 = -\tfrac{1}{2} \qquad B_2 = \tfrac{1}{6}$$

$$B_3 = 0 \qquad B_4 = -\tfrac{1}{30} \qquad B_5 = 0$$

$$B_6 = \tfrac{1}{42} \qquad B_7 = 0 \qquad B_8 = -\tfrac{1}{30}$$

Calcolare un integrale definito • Vediamo ora, con un esempio, come sia possibile ottenere l'integrale definito di una funzione non integrabile con metodi semplici, sostituendola con il suo sviluppo in serie.

Si debba calcolare il seguente integrale definito:

$$\int_0^1 \sin(x^2)dx$$

Sappiamo dal paragrafo precedente che, limitandoci ai primi quattro termini, la funzione $y = sin\ x$ ammette il seguente sviluppo in serie:

$$\sin k = k - \frac{k^3}{3!} + \frac{k^5}{5!} - \frac{k^7}{7!}$$

Se sostituiamo a k il suo valore x^2 otteniamo:

$$\sin(x^2) = x^2 - \frac{x^6}{3!} + \frac{x^{10}}{5!} - \frac{x^{14}}{7!}$$

Se poi andiamo a sostituire lo sviluppo in serie alla funzione di partenza sotto il segno di integrale, avremo:

$$\int_0^1 \sin(x^2)dx =$$

$$= \left(\frac{x^3}{3} - \frac{x^7}{7 \cdot 3!} + \frac{x^{11}}{11 \cdot 5!} - \frac{x^{15}}{15 \cdot 7!} + \cdots \right) \Bigg|_0^1 =$$

$$= \frac{1}{3} - \frac{1}{7 \cdot 3!} + \frac{1}{11 \cdot 5!} - \frac{1}{15 \cdot 7!} + \cdots$$

L'utilizzo dello sviluppo in serie ci ha permesso di trasformare la funzione di partenza in un polinomio facilmente integrabile.

Concetti essenziali

- Un metodo di calcolo si dice diretto se esiste un algoritmo che ne fornisce la soluzione in un numero finito di passi.
- Non sempre un problema matematico ammette una soluzione diretta.
- Un metodo di calcolo si dice iterativo se la soluzione si ottiene per approssimazioni successive, ma in un numero infinito di passi.
- L'uso di un metodo iterativo ha senso solo se la serie è convergente.
- In matematica, si dice convergenza la proprietà di una funzione (o di una successione) di possedere un limite finito al tendere della variabile all'infinito o a un valore determinato.
- Una funzione *f(x)* che sia infinitamente differenziabile nell'intorno di un punto può essere approssimata da una polinomiale, in cui i termini successivi sono ottenuti a partire dalle derivate di ordine crescente della funzione approssimata: tale polinomiale prende il nome di serie di Taylor.
- L'espressione dello sviluppo in serie di Taylor in forma compatta è

$$f(x) = \sum_{n=1}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

- Se nello sviluppo in serie di Taylor $a = 0$, la serie prende il nome di serie di Maclaurin.

Test

1 **Dire se le seguenti affermazioni sono vere o false.**
Un algoritmo che risolve un problema matematico in un numero finito di passi si dice:

- A centrato V F
- B esatto V F
- C convesso V F
- D diretto V F

2 **Dire se le seguenti affermazioni sono vere o false.**
Se il metodo di risoluzione di un problema matematico avviene per approssimazioni successive esso si dice:

- A itinerante V F
- B iterativo V F
- C ittiosauro V F
- D itterico V F

3 **Dire se le seguenti affermazioni sono vere o false.**
Il metodo di Eratostene per la ricerca dei numeri primi è un metodo:

- A interattivo V F
- B diretto V F
- C iterativo V F
- D direzionale V F

4 **Dire se le seguenti affermazioni sono vere o false.**
Il metodo di Taylor permette di:

- A approssimare il valore di un polinomio con una funzione derivabile V F
- B sostituire il valore di una funzione in un punto con una derivata terza V F
- C trovare le derivate di una funzione in un punto V F
- D approssimare il valore di una funzione in un punto con una polinomiale V F

5 **Dire se le seguenti affermazioni sono vere o false.**
Per calcolare il valore di una funzione nel punto $x = 0$ con un metodo iterativo possiamo usare una serie di:

- A Maclaurin V F
- B Macintosh V F
- C Macadam V F
- D Maccluskey V F

6 **Dire se le seguenti affermazioni sono vere o false.**
Un integrale indefinito:

- A è sempre calcolabile attraverso uno sviluppo in serie V F
- B non è calcolabile per via numerica V F
- C si ottiene derivandolo da un insieme di integrali definiti V F
- D viene agevolmente calcolato conoscendo le sue derivate V F

Esercizi

Attenzione: per la soluzione dei seguenti esercizi si faccia riferimento alla documentazione generale del linguaggio R, disponibile online sul sito del corso.

1 **Utilizzando il linguaggio R si scriva un programma che calcoli il valore della funzione:**

$$f(x) = \frac{x}{\sqrt{1-x^2}}$$

per $x = 1/2$
utilizzando i primi tre termini della polinomiale di Taylor.

2 **Utilizzando il linguaggio R, si modifichi il codice relativo all'esercizio precedente in modo da calcolare le diverse approssimazioni con i termini di ordine 1, 2 e 3 della soluzione trovata e si calcoli ad ogni iterazione la differenza tra il valore trovato e quello calcolato per sostituzione diretta.**

3 **Utilizzando il linguaggio R, si scriva un programma che calcoli le successive approssimazioni del valore di:**

$$y = \sin x$$

ricordando che lo sviluppo in serie della funzione in esame è:

$$\sin x = x - x^3/3! + x^5/5! - x^7/7! + ...$$

Si calcoli, per il valore di $x = 0.4$, l'errore rispetto alla soluzione teorica al crescere dell'ordine della polinomiale e si traccino su un grafico la funzione di partenza ed i diversi valori calcolati con le varie polinomiali.

4 **Utilizzando il linguaggio R si scriva un programma che calcoli le prime tre approssimazioni del valore di**

$$f(x) = \cos(\sin x)$$

calcolato con lo sviluppo in serie di Maclaurin nell'intorno del punto $x_0 = 0$.

3 Applicazioni del calcolo numerico

Dopo la trattazione introduttiva dei precedenti capitoli, passiamo ad affrontare uno degli esempi applicativi canonici del calcolo numerico: la determinazione, con diversi metodi, dell'integrale definito di una funzione.

Nei paragrafi che seguono, grazie a un approccio intuitivo e con l'ausilio della grafica, l'allievo prima visualizza e poi arriva a risolvere, attraverso il linguaggio R, questo tipo di problemi per via iterativa, migliorando il risultato attraverso una sempre più precisa approssimazione della soluzione teorica, al crescere del numero delle iterazioni.

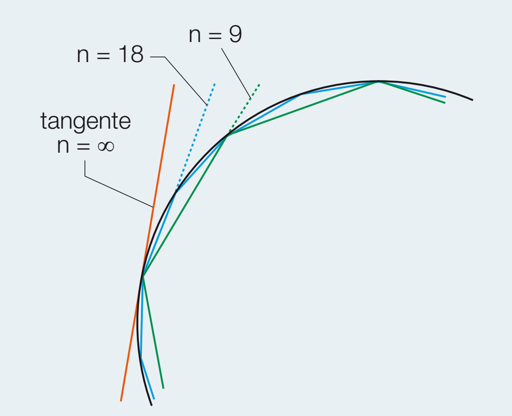
Necessariamente solo introduttivo, il capitolo può essere punto di partenza per alcuni approfondimenti e per verifiche, a livello di codice, di metodi iterativi di altra natura.

1 Parzializzare per risolvere

Come abbiamo più volte ribadito nei paragrafi precedenti, il mondo reale ci pone a confronto con linee, curve, superfici e volumi che non rispecchiano i modelli semplificati e di facile trattazione matematica della geometria euclidea, e che richiedono, per poter essere adeguatamente trattati, l'introduzione di concetti diversi, quale, ad esempio, la geometria frattale.

Allo stesso modo i fenomeni oggetto dell'indagine scientifica assai raramente, o meglio quasi mai, possono essere rappresentati adeguatamente attraverso modelli matematici elementari: i modelli, sempre che esistano, sono frequentemente funzioni di elevata complessità la cui soluzione, per determinate condizioni al contorno, può anche non esistere, oppure richiedere una mole di computo tale da collocarsi al di fuori del rango delle applicazioni pratiche.

Una delle strategie di base - adottata fin dalla più remota antichità in matematica - è quella detta della **discretizzazione**: se una data funzione è complessa e non siamo in grado di trovare un metodo semplice per poterla calcolare, ci accontentiamo di frammentare l'intervallo di nostro interesse in un dato numero di parti all'interno delle quali la funzione di partenza viene sostituita da una sua approssimazione più semplice, o quantomeno affrontabile dal punto di vista computazionale.



Poligoni e cerchio • Un caso classico di questo approccio, che ci proviene dal mondo antico, è la determinazione della lunghezza di una circonferenza a partire dal suo raggio, calcolandola come limite a cui tendono le lunghezze dei perimetri dei poligoni circoscritto e inscritto, al crescere del numero dei lati e, conseguentemente, all'approssimarsi di ogni lato alla tangente alla circonferenza nel punto.

Figura **1**

I metodi iterativi presentano la caratteristica di avvicinarsi progressivamente al valore vero della soluzione: non è detto che qualsiasi problema però possa essere affrontato con un approccio di questo genere.

Un controesempio • Analizziamo insieme, ad esempio, l'andamento della serie:

$$-1, 2, -3, 4, -5, \ldots$$

che può essere espressa in notazione sintetica come

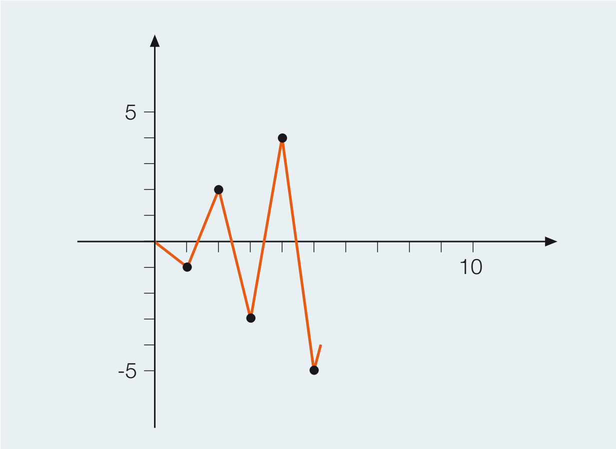
$$(-1)^n \, n$$

Il suo andamento è riportato nel grafico di figura 2.

All'aumentare del numero dei termini, non troviamo un avvicinamento ad un valore limite, ma, al contrario, un'oscillazione di ampiezza sempre maggiore.

Se la serie appena vista fosse il risultato di un tentativo di soluzione attraverso un metodo iterativo, dovremmo ammettere che, anziché migliorare la precisione del nostro risultato aumentando il numero di iterazioni, esso al contrario sarebbe sempre peggiore.

Figura **2**



2 Limiti teorici e valori nella pratica

Vediamo una semplice successione:

$$a_n = \frac{(n+2)}{(2n+1)}$$

Scriviamo un semplice programma in R per valutare il valore a cui tende la successione, sempre che essa converga, ponendo per iniziare il valore n pari a 10:

```
# programma successione 1

N <- 10
A <- 0

#
# qui effettuiamo le operazioni necessarie per arrivare al
# calcolo voluto
#
     A <- (N+2)/(2*N +1)
#
# presentiamo a video N e il risultato
#
print(N)
print(A)
```

Se analizziamo i risultati, vediamo che con

$n = 1$ il risultato è 3
$n = 3$ il risultato è 1

n = 1000 il risultato è 0,5007496
n = 1000 000 il risultato è 0,5000007

Si tratta, infatti, di una sequenza convergente il cui valore limite è 0,5.
In modo del tutto analogo, la successione:

$$a_n = \frac{n}{n+1}$$

per crescenti valori di n assume i valori:

$a_1 = 1/2$
$a_2 = 2/3$
$a_3 = 3/4$
$a_4 = 4/5$
...
$a_{10} = 10/11$

e così via.
Eseguendo per valori crescenti di n il semplice codice:

```
# programma successione 2

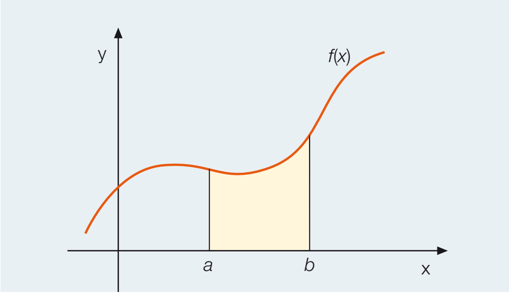
N <- 10
A <- 0

#
# qui effettuiamo le operazioni necessarie per arrivare al
# calcolo voluto
#
     A <- N/(N +1)
#
# presentiamo a video N e il risultato
#
print(N)
print(A)
```

ci accorgiamo che la successione converge e che il suo limite teorico è 1.
Il valore effettivo del limite della successione però si raggiungerebbe solo con un numero infinito di termini: questo comporterebbe un tempo di esecuzione infinito, quindi inaccettabile per qualsiasi applicazione nel mondo reale.

3 Calcolo degli integrali definiti

Figura **3**



Una delle applicazioni classiche del metodo numerico è la ricerca di soluzioni, per approssimazioni successive, del valore dell'integrale definito di una funzione $f(x)$, espresso convenzionalmente con la notazione:

$$\int_a^b f(x)dx$$

Da un punto di vista geometrico, sappiamo che l'integrale definito di una funzione, in un dato intervallo, corrisponde al valore

dell'area compresa tra il grafico della funzione e l'asse delle ascisse, in quel dato intervallo (figura 3).

Formula fondamentale • In alcuni casi, quelli in cui è semplice reperire la primitiva della funzione $f(x)$ sotto segno di integrale, se:

$$F'(x) = f(x)$$

allora vale la formula fondamentale del calcolo integrale:

$$\int_a^b f(x)dx = F(b) - F(a)$$

Il calcolo dell'integrale si riduce, in questo caso, a una semplice sostituzione dei valori che la primitiva assume agli estremi dell'intervallo.

Più interessante, e molto più comune, il caso in cui la determinazione della primitiva non è immediata o, addirittura, non è ottenibile: l'integrale richiesto viene allora calcolato in termini di area sottesa, come somma di superfici approssimanti: operazione che può essere eseguita con metodi diversi, come andiamo a esaminare nei prossimi paragrafi.

Natura della soluzione • Notiamo che la soluzione di un integrale definito non è più una funzione della variabile x, se stiamo operando con integrali di funzioni di una sola variabile, ma è un numero, cioè una costante.

Metodo dei rettangoli • Nelle figure che seguono, notiamo che la superficie dell'area sottesa dalla funzione può essere approssimata come somma di rettangoli, tutti

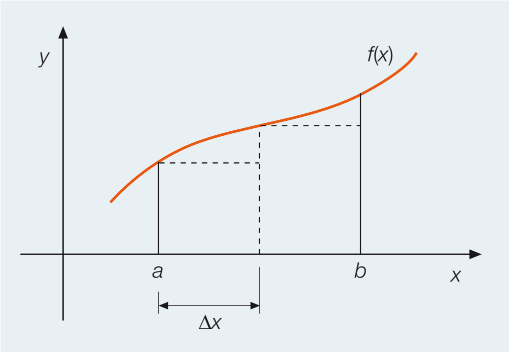


Figura **4**

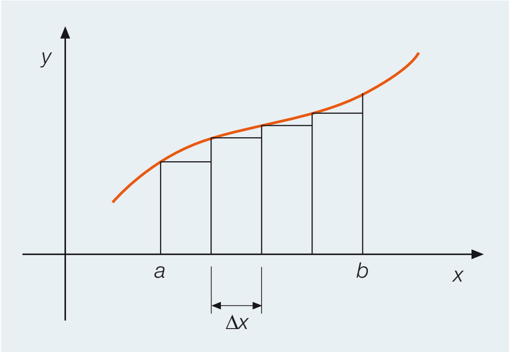


Figura **5**

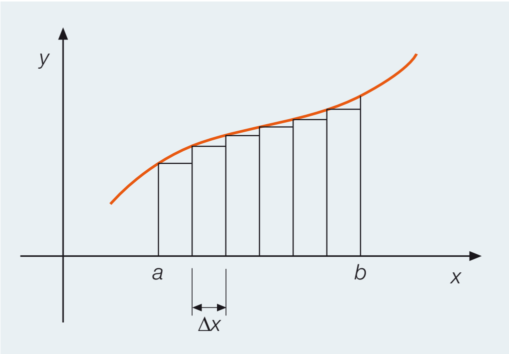


Figura **6**

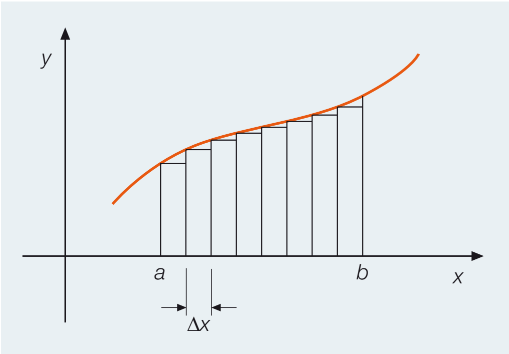


Figura **7**

di pari larghezza Δx, la cui altezza varia in funzione del valore della funzione in uno dei loro estremi.

Notiamo che, in tutti i casi, come altezza di ogni rettangolo è stato assunto il valore che $f(x)$ assume in corrispondenza dell'estremo sinistro dell'intervallo Δx.

La scelta dell'estremo sinistro, rispetto a quello destro, è puramente soggettiva: per la validità del metodo basta che, una volta scelto un criterio (estremo sinistro o estremo destro), esso venga conservato per tutti i rettangoli che andranno ad approssimare la funzione.

Nelle figure che seguono, vediamo come la stessa funzione $f(x)$ venga approssimata da rettangoli in cui il valore delle altezze sia calcolato sull'estremo destro dell'intervallo Δx.

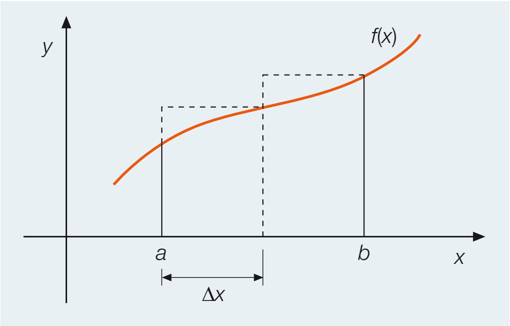


Figura **8**

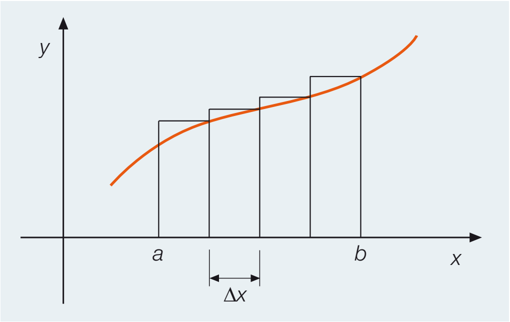


Figura **9**

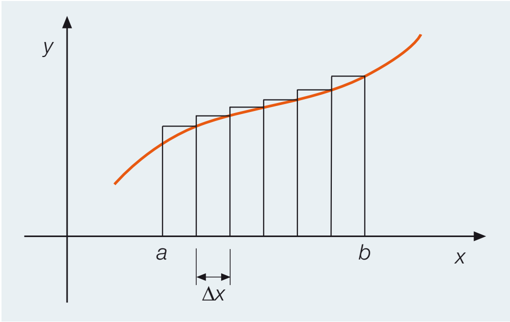


Figura **10**

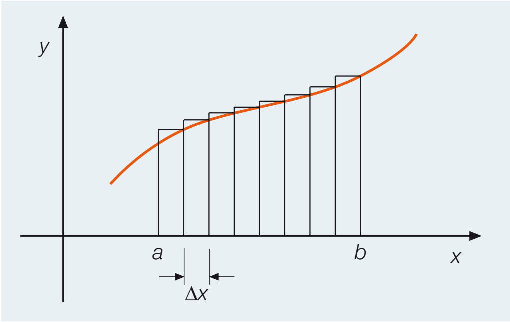


Figura **11**

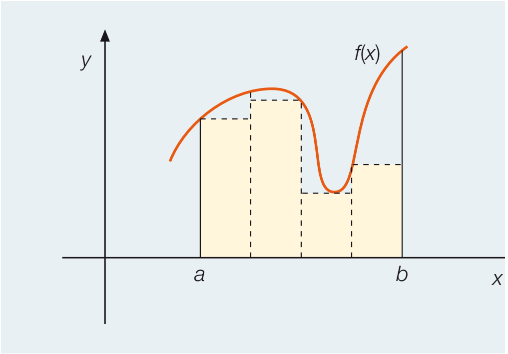
Possiamo notare che, al diminuire dell'ampiezza di Δx, quindi all'aumentare del numero dei rettangoli approssimanti, la somma delle aree dei rettangoli e l'area compresa tra $f(x)$ e l'asse delle ascisse tendono a coincidere. Si tratta, infatti, di un metodo convergente.

Un programma per il calcolo • Cerchiamo ora di costruire, aiutandoci con considerazioni generali e con un diagramma di flusso, un programma in grado di calcolare l'integrale definito di una funzione $f(x)$, usando il metodo della somma dei rettangoli appena visto.

Per comodità, considereremo rettangoli la cui altezza sia calcolata sull'estremo sinistro di ogni intervallo Δx.

Scaloidi • Le figure a gradini che sono ottenute dall'affiancamento dei rettangoli costruiti come abbiamo appena visto prendono il nome di **scaloidi**.

Possiamo assumere altri criteri per la definizione dell'altezza dei rettangoli: decidere, ad esempio, che per ogni intervallo Δx, l'altezza h del rettangolo venga data dal valore minimo assunto da $f(x)$ in Δx. In questo caso avremo uno *scaloide inscritto*: mentre se assumessimo come altezza del rettangolo il valore massimo di $f(x)$ nell'intervallo considerato, avremmo uno *scaloide circoscritto*.



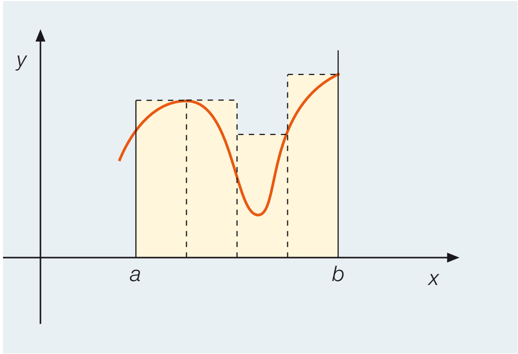


Figure **12** e **13**

Considerazioni • Per prima cosa, dovremo decidere in quante parti dividere l'intervallo (a,b) dato. Come sappiamo, un buon codice è quanto più possibile indipendente dai valori specifici di un problema, per cui supporremo che l'intervallo sia suddiviso in n parti: ci sarà utile, per verificare la convergenza del metodo, far assumere a n parti valori diversi, in fase di collaudo.

Divideremo, quindi, l'intervallo (a,b) in n parti, per cui i punti in cui dovremo calcolare il valore della funzione saranno quelli di ascisse pari a:

$x_0 = a$
$x_1 = a + \Delta x$
$x_2 = a + 2\Delta x$
$x_3 = a + 3\Delta x$
. . .
$x_n = a + n\Delta x = b$

Ogni rettangolo avrà area pari a (*base* x *altezza*). Essendo l'altezza h calcolata sull'estremo sinistro di Δx, detto x_i, l'area di ogni rettangolo sarà:

$$Area_i = \Delta x \times f(x_i)$$

quindi, l'area totale dei rettangoli tracciati sarà data dalla somma delle aree di tutti i rettangoli:

$$\sum_{i=0}^{n-1} f(x_i)\ \Delta x$$

Listato in linguaggio R • Ora che il metodo è stato illustrato, non resta che scrivere un breve programma in linguaggio R che esegua le operazioni richieste in un intervallo dato.

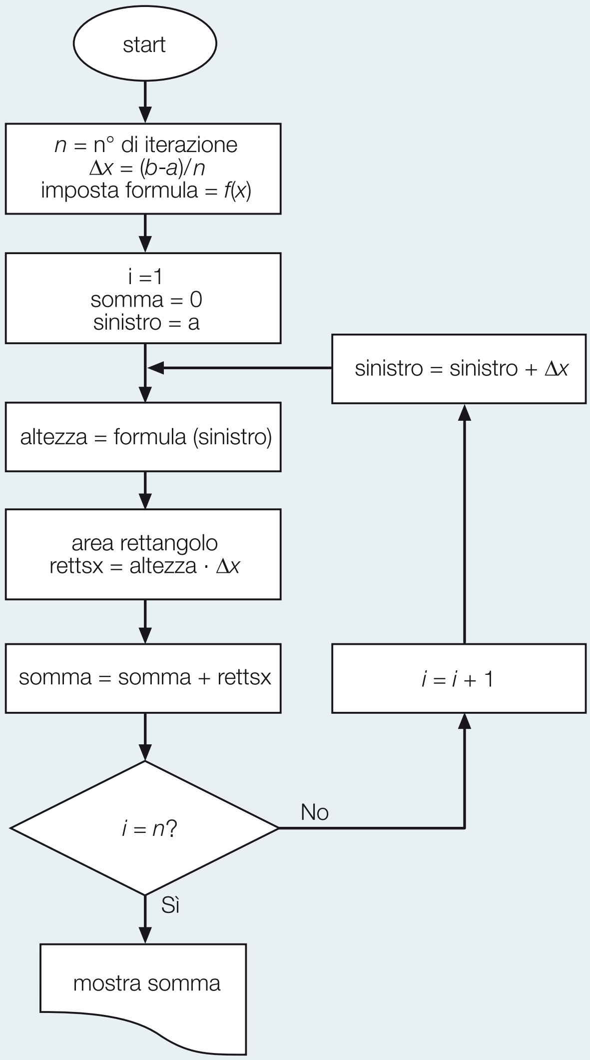
Per semplicità, inseriremo direttamente nel codice la funzione *f(x)* e l'intervallo (a, b), che, per gli scopi di questo esempio, assumeremo pari a:

$$f(x) = x^2 - 3x + 5 \qquad a = 4 \qquad b = 8$$

Qui di seguito troviamo il listato del programma e il relativo schema di flusso (fig. 14).

```
# Programma per il calcolo dell'integrale col metodo
# degli scaloidi, con rettangoli appoggiati a sinistra

# Vogliamo trovare un integrale definito
# ad esempio y = x^2 - 3x +5, con a = 4 e b = 8
#
# impostiamo il numero di intervalli in cui suddividere (a,b)
```




```
# ad esempio 4
#
n <- 4
#
# assegniamo i valori di a e b
#
a <- 4
b <- 8

# calcoliamo il DeltaX, dividendo in n parti,
cioè 4, il segmento (a,b)
#
deltax <- (b-a)/n
#
# qui reiteriamo le  operazioni necessarie
per arrivare al calcolo voluto
#
xcorrente <- a
ycorrente <- a^2 - 3*a +5
arearettangolo <- 0
areatotale <- 0
#
#ycorrente è l'altezza del rettangolo
#
for(i in 1:n) {
      arearettangolo <- (deltax * ycorrente)
      areatotale<-areatotale+arearettangolo
#
# calcolo nuova coordinata sulle x
#
xcorrente <- xcorrente + deltax
#
# calcolo nuova altezza di y(x) con x =
xcorrente
#
ycorrente <- xcorrente^2 - 3* xcorrente +5
      }
#
# presentiamo il risultato
#
print(areatotale)
```

Figura **14**

Verifica della convergenza • Proviamo a far girare il programma con valori via via crescenti del termine n parti: questo significa utilizzare rettangoli sempre più stretti per "ricoprire" l'area compresa tra la curva della funzione e l'asse delle ascisse.

Con $n = 4$, avremo:

```
> print(areatotale)
[1] 80
>
```

con n = 10, avremo:

```
> print(areatotale)
[1] 90.24
>
```

e passando a n = 100, troveremo:

```
> print(areatotale)
[1] 96.6144
>
```

4 Un metodo non lineare

Dopo aver analizzato con attenzione il metodo sopra esposto, possiamo ribadire che, in realtà, la scelta dell'altezza del rettangolo può essere effettuata seguendo criteri diversi. Possiamo anche approssimare una funzione nel modo mostrato in figura 15.

In questo caso, l'altezza del rettangolo è data dal valore della funzione nel punto medio dell'intervallo Δx considerato.

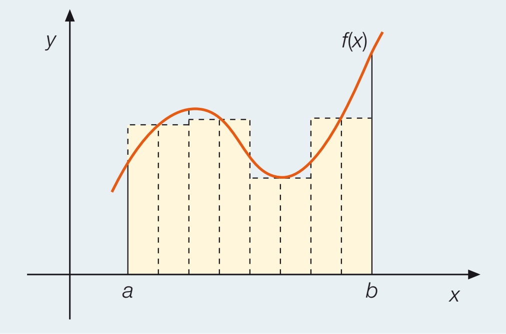


Figura **15**

Metodo dei trapezi • Un altro metodo utilizzato sin dal XVII secolo per la risoluzione numerica del calcolo degli integrali definiti è quello detto "**metodo dei trapezi**": in luogo di costruire rettangoli approssimanti, si costruiscono trapezi, come illustrato nella figura 16.

L'area di ogni trapezio è data da:

[(*base maggiore* + *base minore*) x *h*)] / 2

in cui h, cioè l'altezza di ogni trapezio, è sempre il consueto valore Δx di suddivisione degli intervalli.

Si noti che, in linea teorica, nessuno impedisce di operare con intervalli Δx di ampiezza diversa, anche se per comodità si opera nella pratica sempre mantenendo costante l'altezza delle figure, rettangoli o trapezi che siano.

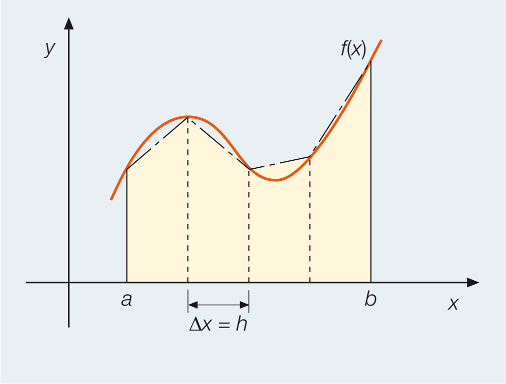


Figura **16**

Diagramma di flusso • Di seguito il listato in linguaggio R, relativo al metodo dei trapezi. Nella fig. 17 il relativo diagramma di flusso.

```
# Programma per il calcolo dell'integrale col metodo
# dei trapezi

# Vogliamo trovare un integrale definito
# ad esempio y = x^2 - 3x +5, con a = 4 e b = 8
#
# impostiamo il numero di intervalli in cui suddividere (a,b)
# ad esempio 4
#
n <- 4
#
```

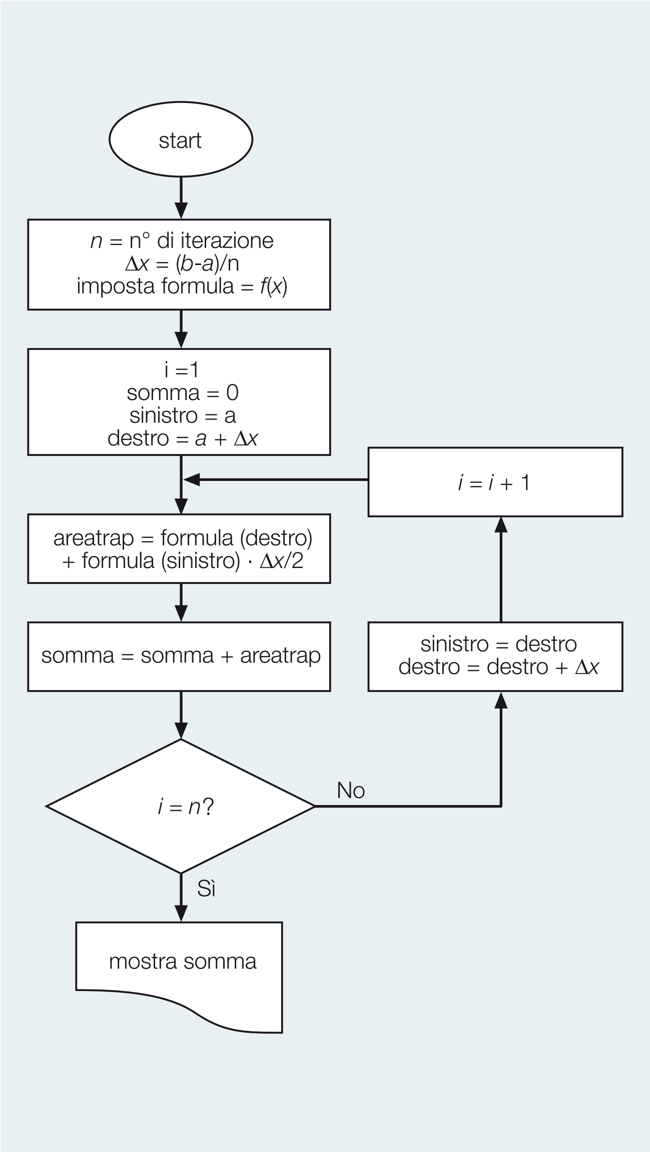


Figura **17**

```
# assegniamo i valori di a e b
#
a <- 4
b <- 8

# calcoliamo il DeltaX, dividendo in n
parti, cioè 4
# il segmento (a,b)
deltax <- (b-a)/n
#
# qui reiteriamo le  operazioni necessarie
per arrivare al
# calcolo voluto
#
xcorrente <- a
ycorrente <- a^2 - 3*a +5
xprossimo <- a + deltax
yprossimo <- xprossimo^2 - 3*xprossimo +5
areatrapezio <- 0
areatotale <- 0
# ycorrente è l'altezza del rettangolo
for(i in 1:n) {
          areatrapezio  <-  ((ycorrente  +
yprossimo)*deltax)/2
          areatotale   <-   areatotale   +
areatrapezio
# calcolo nuova coordinata sulle x
          xcorrente <- xprossimo
          xprossimo <- xcorrente + deltax
# calcolo nuova coordinata sulle y
          ycorrente <- yprossimo
          yprossimo   <-   xprossimo^2   -
3*xprossimo +5
    }
#
# presentiamo il risultato
#
print(areatotale)
```

Approssimazioni lineari e non • Osservando da un punto di vista geometrico le soluzioni finora viste, notiamo come, in realtà, in ognuno dei metodi analizzati sia stata operata una semplificazione. Di fatto, in ogni intervallo Δx, in luogo dell'andamento più o meno complesso di $f(x)$ sul piano x,y si è sostituito un segmento di retta.

Nel caso del metodo dei rettangoli, ogni segmento è sempre parallelo all'asse delle ascisse, quindi il valore y dell'ordinata è costante. Al contrario, il metodo dei trapezi sostituisce alla funzione $f(x)$ nell'intervallo Δx il segmento di retta congiungente i due valori di $f(x)$ agli estremi dell'intervallo stesso.

Sia in un caso sia nell'altro, quindi, la funzione originaria viene di fatto assimilata a un andamento lineare, anche se di tipo diverso.

Metodo di Simpson • Un tipo diverso di approssimazione dell'area sottesa da una funzione $f(x)$ è quello utilizzato nel metodo che prende il nome di ***Metodo di Simpson***: anziché considerare segmenti di retta approssimanti, cioè funzioni di ordine 1, l'approssimazione

avviene attraverso funzioni quadratiche, cioè di ordine 2.

In termini più semplici: anziché approssimare la funzione $f(x)$ con segmenti di retta, si costruisce alla prima iterazione una parabola, passante per i tre punti a, b e $(b-a)/2$ (cioè il punto di mezzo del segmento ab), come in figura 18.

Alla successiva iterazione, effettuata prendendo come nuovi intervalli Δ1 e Δ2, si avranno due parabole, ognuna delle quali sarà una migliore approssimazione della curva descritta dalla $f(x)$ originaria.

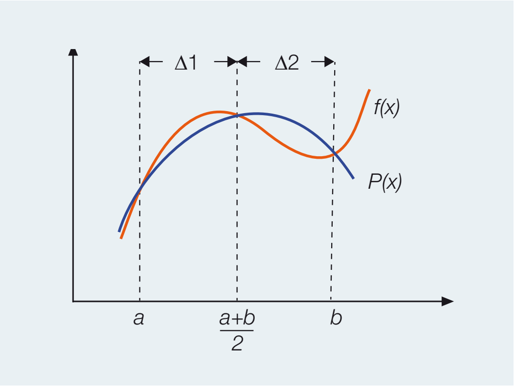


Figura **18**

Al crescere del numero delle iterazioni, la somma delle aree tenderà sempre più a convergere al valore dell'integrale definito in esame.

5 Applicazioni pratiche

Potrebbe sembrare, a prima vista, che i problemi affrontati siano interessanti ma che, fondamentalmente, si tratti di esercizi teorici di tipo matematico, di fatto senza riscontri nella vita pratica di tutti i giorni.

Certamente, è molto improbabile che chi svolge un lavoro semplice o manuale abbia la necessità di calcolare integrali definiti, ma vorremmo concludere questo capitolo mostrando alcuni dei campi di applicazione più frequenti per i metodi incontrati e facendo alcune considerazioni finali.

Alcuni esempi • Un semplice caso di applicazione è quello della determinazione dell'area di un terreno dal perimetro complesso, come quello riportato in figura 19A.



Figura **19**

Non avremo, in questo caso, una funzione analitica $f(x)$ da integrare, ma, se il topografo ha rilevato correttamente i punti salienti del perimetro, avremo una serie di coordinate che potremo utilizzare come valori (x, y) ed ottenere l'area come differenza tra la misura della superficie sottesa dalla parte superiore del perimetro e quella inferiore, come in figura 19B.

Il metodo ha trovato storicamente applicazioni in numerose branche della fisica, ad esempio per la determinazione dei baricentri di solidi la cui sezione non sia esprimibile con funzioni elementari, nei problemi di idrodinamica e nell'analisi del comportamento dei corpi galleggianti, per non citare che i casi più diffusi.

Le possibilità dei linguaggi orientati al calcolo • Dopo aver imparato a costruire il codice necessario per il computo degli integrali definiti, con diversi metodi, è

importante ricordare come linguaggi creati appositamente per la soluzione di problemi di natura matematica e scientifica, come R o MATLAB®, mettano a disposizione del programmatore strumenti già pronti per la descrizione delle funzioni di una o più variabili. Avendo a disposizione dichiarazioni standard e funzioni interne già pronte, oppure librerie acquisibili separatamente, il programmatore non deve più preoccuparsi di scrivere il codice, ma può limitarsi a effettuare, dall'interno della propria procedura, una chiamata a una funzione che restituisce il valore dell'integrale definito cercato.

Naturalmente, l'esistenza di questi metodi non sposta minimamente il problema della determinazione della loro applicabilità e quella della valutazione degli errori commessi dalla procedura di approssimazione, in funzione del numero delle iterazioni prescelto.

La funzione integrate • Specificatamente, R mette a disposizione una funzione

```
integrate
```

attraverso la quale è possibile, con una semplice chiamata, ottenere il calcolo dell'integrale definito di una funzione in una sola variabile (per maggiori dettagli, si rimanda alla documentazione del linguaggio).

La funzione `integrate` accetta anche dei parametri che esprimono i valori di tolleranza sull'errore commesso attraverso l'approssimazione intrinseca del metodo.

Concetti essenziali

- Attraverso la discretizzazione, la soluzione a un problema complesso può essere ricondotta, in determinate situazioni, a quella di problemi più semplici, a patto di accettare una certa imprecisione del risultato finale.
- Se l'errore che si compie al crescere del numero di iterazioni diviene sempre più piccolo in valore assoluto, si dice che la soluzione converge al risultato cercato, anche se sovente lo raggiungerà solo con un numero infinito di passaggi.
- Nelle applicazioni pratiche non è possibile operare con precisione assoluta: i diversi ambiti applicativi accettano, quindi, approssimazioni che sono funzione della natura del problema, della tipologia di soluzione ritenuta adottabile e dell'ordine di grandezza degli strumenti e dei metodi con cui si eseguono le misure in un dato ambito.
- Il problema generale della determinazione dell'integrale definito di una funzione $f(x)$ su un intervallo (a, b) può essere ricondotto al calcolo di una serie di superfici approssimanti - per difetto o per eccesso - porzioni dell'area compresa fra la curva data e l'asse delle ascisse.
- Il metodo risolutivo attraverso la somma dei rettangoli può essere computato calcolando l'altezza di ogni elemento rispetto all'estremo sinistro, a quello destro o al punto medio di ogni intervallo Δx.
- Il metodo dei trapezi approssima la funzione data nell'intervallo Δx attraverso una retta congiungente i punti di valore $y_n = f(x_n)$ e $y_{n+1} = f(x_{n+1})$.
- Attraverso il metodo cosiddetto di Simpson l'approssimazione avviene utilizzando una funzione quadratica, tipicamente una parabola passante per i tre punti notevoli (il valore assunto dalla funzione nei due estremi e nel punto medio) degli elementi in cui viene diviso l'intervallo (a, b).

Test

1 **Dire se le seguenti affermazioni sono vere o false.**
Quando un calcolo è eseguibile, il suo risultato:
- **A** è sempre esatto **V F**
- **B** può essere approssimato **V F**
- **C** deve sempre essere approssimato **V F**
- **D** talvolta richiede degli aggiustamenti a mano **V F**

2 **Dire se le seguenti affermazioni sono vere o false.**
Al crescere del numero delle iterazioni di un algoritmo, il risultato:
- **A** migliora se l'algoritmo converge **V F**
- **B** peggiora se l'algoritmo converge **V F**
- **C** se tende a zero è più preciso, ma non sempre **V F**
- **D** dopo un certo numero di iterazioni finisce sempre col deviare dal valore esatto **V F**

3 **Dire se le seguenti affermazioni sono vere o false.**
Se un algoritmo non porta alla determinazione di un valore esatto in un numero finito di passi:
- **A** non serve a niente **V F**
- **B** può essere utile se la soluzione è accettabilmente precisa **V F**
- **C** a ogni passo dobbiamo calcolare l'errore compiuto **V F**
- **D** è utile valutare se il guadagno in precisione al crescere delle iterazioni è significativo per la nostra applicazione **V F**

4 **Dire se le seguenti affermazioni sono vere o false.**
Il calcolo di un integrale indefinito:
- **A** dipende dalla precisione che si vuole ottenere nel risultato **V F**
- **B** va affrontato con il metodo dei trapezi **V F**
- **C** è facilmente risolubile con il metodo dei rombi **V F**
- **D** non può essere affrontato per via numerica **V F**

5 **Dire se le seguenti affermazioni sono vere o false.**
Lo scaloide risultante dall'approssimazione della funzione $f(x)$ nella risoluzione con il metodo dei rettangoli del suo integrale definito nell'intervallo (a, b), nei confronti della superficie esatta cercata, è:
- **A** sempre approssimante per eccesso **V F**
- **B** in alcuni punti approssimante per eccesso ed in altri per difetto **V F**
- **C** sempre approssimante per difetto **V F**
- **D** oscillante in funzione dei valori di a e b **V F**

6 **Dire se le seguenti affermazioni sono vere o false.**
Il metodo dei trapezi nella risoluzione per via numerica dell'integrale definito di una funzione $f(x)$ è:
- **A** l'applicazione di una approssimazione attraverso funzioni lineari a tratti **V F**
- **B** un metodo che converge solo per un numero n di iterazioni > 100 **V F**
- **C** un'approssimazione attraverso una polinomiale di grado $n > 3$ **V F**
- **D** identico al metodo dei rettangoli, differisce solo per il calcolo del punto intermedio del segmento Δx su cui si calcola l'altezza della figura geometrica approssimante l'area cercata **V F**

Esercizi

Attenzione: per la soluzione dei seguenti esercizi si faccia riferimento alla documentazione generale del linguaggio R, disponibile nel sito Web collegato al presente volume.

1 **Utilizzando il linguaggio R si scriva un programma, ben commentato, che calcoli il seguente integrale definito:**

$$y = \int_a^b f(x)dx$$

con $f(x) = (\sin x + \cos x)$, $a = 0{,}3$ e $b = 2$, attraverso il metodo dei rettangoli appoggiati a destra, con 5 iterazioni.

2 **Riprendendo l'esercizio precedente, utilizzando il linguaggio R, si modifichi il codice precedente in modo da calcolare il valore approssimato con 5, 10, 50 e 100 iterazioni.**
Si commentino i risultati.

3 **Utilizzando il linguaggio R si scriva un programma che calcoli le successive approssimazioni del valore di:**

$$\int_0^{1/2} \sin x - 2\sin 2x + \cos x \, dx$$

con 8 iterazioni, scrivendo a video gli 8 successivi risultati.

Si traccino sullo schermo il grafico della funzione integranda e quello delle prime tre approssimazioni, con colori diversi.

4 **Riprendendo il problema al punto 3, si modifichi il codice per accettare da tastiera il numero *n* delle iterazioni richieste, rifiutandolo con un messaggio se è superiore a 100 ed accettando solo valori ≥ 2.**

5 **Si provi a immaginare se esiste una funzione $y = f(x)$ tale per cui, indipendentemente dal numero delle iterazioni, il risultato della regola dei trapezi per la**

risoluzione di

$$y = \int_a^b f(x)dx$$

è sempre esattamente uguale al valore effettivo dell'area compresa tra la curva $f(x)$ e l'asse delle ascisse.

6 **Utilizzando il linguaggio R si scriva un programma che permetta all'utente di scegliere, da tastiera, quale metodo usare per la risoluzione dell'integrale definito seguente:**

$$y = \int_{10}^{20} \left(2\sin x - \frac{\cos 2x}{2} \right) dx$$

dando la scelta tra metodo dei rettangoli a sinistra, quello dei rettangoli a destra, metodo dei rettangoli calcolati sul punto medio di Δx e metodo dei trapezi.
Il programma chieda anche interattivamente all'operatore il numero di iterazioni che si desidera effettuare e mostri a video il risultato ottenuto.

Sezione **D**

Informatica per il sapere

D1 Condividere le conoscenze

D2 Informatica per collaborare

D3 Modelli e popolazioni

1 Condividere le conoscenze

In questa sezione vengono presentati due degli strumenti informatici più comunemente usati per la condivisione delle informazioni via Web: il blog e il wiki.

Sono introdotti i concetti di base che informano questo tipo di comunicazione, con una particolare attenzione agli scopi e allo spirito con cui essi sono stati creati e con cui dovrebbero sempre essere utilizzati. La parte operativa, con gli esempi di creazione di un proprio blog personale, è rimandata al sito Web collegato al presente volume. Analizzeremo in questo capitolo Wikipedia, il wiki più famoso.

L'apprendimento collaborativo può trovare nei blog e nei wiki un ausilio potente: le pagine che seguono introducono le generalità che serviranno di fondamento alle applicazioni pratiche degli strumenti ad altre discipline, oggetto dei capitoli seguenti.

1 Informazione e collettività

Conoscere e condividere • Sia che una singola persona si accinga a studiare un determinato problema - non importa se di natura scientifica o meno - sia che coloro che partecipano a questa ricerca siano più d'uno, soprattutto se lontani in termini di tempo e di spazio, risulta indispensabile avere a disposizione un contenitore in cui immagazzinare i materiali su cui si opera e i risultati, anche parziali, del lavoro, in modo da renderli disponibili, almeno all'interno del gruppo.

La soluzione potrebbe essere quella di tenere dei log files - ne abbiamo parlato nei primi anni di questo corso - cioè una sorta di diario di bordo in cui ogni partecipante al gruppo di lavoro, giorno dopo giorno, si impegna a registrare tutto quello che avviene, prendendo nota di fatti, scoperte, comunicazioni e avvenimenti.



Figura **1**
Penna e calamaio

Dato che viviamo in un'epoca in cui la tecnologia è alla portata di tutti, difficilmente immagineremmo il nostro ipotetico studioso alle prese con un grosso registro e con una penna d'oca intinta nel calamaio: più probabilmente, userebbe il suo Personal Computer e un elaboratore di testi.

Un diario in rete • Se chi redige un log file si proponesse semplicemente di scrivere la cronistoria, magari legata a una tipologia di eventi che si esauriscono nel tempo, quali potrebbero essere i risultati delle partite di calcio di un campionato o l'elenco delle manifestazioni della propria città in un certo arco temporale, un semplice file di Word, magari arricchito con qualche immagine, potrebbe rappresentare una soluzione facile ed economica. Se invece l'obiettivo dell'autore fosse quello di condividere le sue idee e le sue esperienze con altri, potrebbe renderle accessibili in rete, pubblicandole in un blog e, se lo desidera, limitandone l'accesso a una lista di utenti selezionati.

Blog e blogger • Un blog è un esempio tipico di comunicazione fondamentalmente monodirezionale. Di cosa si tratta?

Per prima cosa, il termine ***blog*** è l'abbreviazione di quella che fu la sua denominazione iniziale, cioè *weblog*, termine composto da due parole: *Web* e *log* (come abbiamo precedentemente detto, per log si intende il "giornale di bordo", cioè una registrazione di eventi in ordine cronologico).

Con un gioco di parole, dal termine weblog vennero estratte per scherzo le due parole "*we*" e "*blog*": cioè, "*noi blogghiamo*", attribuendo a questa espressione il significato di "*noi comunichiamo attraverso un blog*" cioè comunichiamo attraverso un weblog.

Figura **2**
La pagina di un blog.



Il termine blog era un neologismo, ma piacque immediatamente, al punto che il vecchio termine *weblog* passò in disuso, sostituito dal più agile *blog*.

Un blog è una modalità di comunicazione fondamentalmente monodirezionale: il titolare del blog vi inserisce i suoi contributi, che possono poi essere consultati dai soggetti che hanno accesso al sistema.

Attraverso un apposito strumento software, il proprietario del blog può in qualsiasi momento accedere per inserire dei contenuti, che rimarranno visibili in ordine cronologico.

Gli interventi su un blog sono detti in gergo ***post***, e compaiono in ordine cronologico inverso: il primo visualizzato sarà sempre il più recente.

L'argomento del blog può essere qualsiasi: ogni ***blogger*** (si chiama così il proprietario del blog) è libero di decidere di cosa parlare.

In alcuni blog i lettori possono inserire dei commenti: a volte, tali contributi vengono pubblicati direttamente, senza attività di filtro e verifica da parte del titolare del blog, mentre in altri casi la loro pubblicazione è sottoposta ad un controllo e ad una sorta di approvazione. La persona - o le persone - che svolgono questo compito di controllo e validazione dei commenti a monte della loro messa in rete prendono il nome di ***moderatori***.

Vi sono blog in cui una mamma racconta le proprie avventure e le proprie esperienze di vita quotidiana; altri in cui uno sportivo parla degli allenamenti, delle scelte di materiali tecnici o di regimi dietetici e dei risultati ottenuti; altri ancora in cui uomini politici presentano le proprie idee e le proprie posizioni sui fatti del giorno o su temi di attualità.

Speakers' Corner • Nel celebre Hyde Park, uno dei più bei giardini pubblici londinesi, esiste da secoli uno spiazzo, denominato



Figura **3**
Lo Speakers' Corner in Hyde Park.

Speakers' Corner, cioè l'*angolo degli oratori*. In esso, chiunque ritenga di avere qualcosa da dire può prendere la parola ed esprimersi nella più totale libertà. I passanti e i curiosi si fermano ad ascoltare - se sono interessati - e se non lo sono proseguono nella propria passeggiata. Nessuno proibisce all'oratore di parlare, ma nessuno obbliga i presenti ad ascoltare: non si tratta di una predica, e neppure l'uditorio si sente obbligato, dall'educazione o dalle circostanze, a rimanere sul posto ad ascoltare per tutto il tempo dell'evento.

Dobbiamo immaginare un blog come qualcosa di molto simile allo Speakers' Corner: ognuno è libero di raccontare, così come ognuno è libero di ascoltare - in questo caso di leggere - ma nessuno, in nessun modo, può essere costretto ad accedere a un blog altrui né tantomeno a rimanere nella pagina, se non la ritiene in qualche modo interessante.

In un blog i contenuti cambiano solo nel senso in cui nel tempo ai primi si aggiungono i successivi: ripercorrendo all'indietro l'elenco dei post, li si ritrova come essi erano stati inseriti originariamente, anche se nessuno proibisce al blogger di scrivere post successivi in cui si vadano a modificare informazioni od opinioni espresse in interventi precedenti.

Blogosphere • Con il termine ***blogosphere*** (*blogosfera*) si intende la comunità virtuale di tutti i blog e di tutte le loro correlazioni. Questo termine evidenzia l'aspetto di ***social network*** dei blog, che vengono visti come canale di comunicazione a elevato impatto potenziale.

Figura **4**
La testata di un blog professionale.



Dimensioni del fenomeno • Per renderci conto della diffusione di questo metodo di comunicazione, si pensi che nel 2012 sono stati consultati ogni mese, nel mondo, oltre 500 milioni di pagine blog! E questo senza contare gli accessi legati ai social network più diffusi, come Facebook, all'interno dei quali gli utenti possono di fatto creare una sorta di piccolo blog personale, che può essere anche arricchito con immagini, link a video, ecc.

Blog amatoriali e blog professionali • Costruire un proprio blog è estremamente semplice: sono disponibili in rete strumenti gratuiti che permettono anche a utenti che non conoscono la programmazione Web di realizzare in breve tempo il proprio blog. Grazie a un ricco repertorio di pagine "pronte all'uso", graficamente piacevoli, una famiglia può raccontare ad amici e parenti lontani i fatti quotidiani, oppure un appassionato di una qualche disciplina potrà narrare dei libri che sta leggendo, delle scoperte che sta facendo o di come intende organizzare le sue prossime attività.

L'interesse del pubblico per un blog è legato sia all'argomento trattato, sia all'autorevolezza o alla fama del suo autore. È d'altro canto essenziale che il blog sia mantenuto vivo: se da troppo tempo non viene più scritto nulla di nuovo, il blog in qualche modo "muore" - come si dice in gergo - e perde di attrattiva per coloro che lo frequentavano.

Un blog può anche essere un importante strumento per aziende e attività di ogni tipo, commerciali, culturali, sociali e addirittura accademiche: esso può essere gestito in modo altamente professionale e diventare un punto di riferimento per una comunità interessata ai contenuti che in esso vengono presentati.

2 Spazi per condividere e collaborare

Un autore, tanti autori • Se il blog è un canale di comunicazione sostanzialmente monodirezionale, la situazione cambierebbe completamente se dovessimo gestire i contributi di più autori. Il primo problema sarebbe quello di avere uno spazio condiviso in scrittura, non solo in lettura: il secondo punto da risolvere sarebbe quello di stabilire delle regole per chi intenda contribuire.

Il forum • Una delle possibilità offerte dal Web è il ***forum***: si tratta di uno spazio virtuale il cui nome richiama la piazza delle antiche città romane, detta appunto il Foro, in cui i cittadini si incontravano per scambiarsi idee, notizie ed opinioni.

Solitamente un forum è dedicato ad un particolare argomento: al suo interno esistono poi suddivisioni che corrispondono ad altrettanti temi, sui quali tutti i partecipanti possono esprimere le proprie opinioni. Gli iscritti al forum possono solitamente anche creare nuovi argomenti, che prendono il nome di ***thread*** (in inglese *filo*, nel senso di *filo del discorso*).

Non è questa la sede per approfondire l'argomento, ma è necessario dedicare ancora qualche osservazione alle regole generali di questo tipo di strumenti.

Regole di comportamento • Si dovrà fare in modo che gli interventi siano pertinenti, giustificati, frutto di effettivo apporto di ulteriore conoscenza e non solo della mania di protagonismo di chi pretende di far valere la propria opinione in quanto tale e non perché effettivamente meritevole di ascolto.

Come in qualsiasi consesso civile, quando siamo in più d'uno a voler parlare è indispensabile condividere delle regole, semplici ma ineludibili:

- si deve parlare uno alla volta
- si deve accettare l'opinione di chi ne sa (dimostrabilmente) più di noi
- non si deve interrompere chi sta parlando

- non si può monopolizzare l'uditorio parlando in continuazione
- se ci sono punti dibattuti, si deve se necessario porre un punto fermo alla discussione, anche magari dichiarandola aperta in attesa di migliori argomentazioni
- non si possono insultare od offendere i partecipanti
- non si possono minacciare i partecipanti
- non si possono obbligare gli altri ad assumere il nostro punto di vista solo in virtù del fatto che siamo noi a sostenerlo, o qualcuno o qualcosa in cui crediamo

Negli ultimi due o tre decenni, la condivisione delle conoscenze all'interno delle realtà produttive e, soprattutto, dei gruppi di ricerca, ha potuto svilupparsi decisamente con l'avvento delle reti e delle applicazioni di rete. La rete, infatti, rende lo scambio di informazioni istantaneo e indipendente dalla distanza geografica.

Già da una ventina d'anni esistono applicazioni di rete che facilitano lo scambio di informazioni tra soggetti diversi: la posta elettronica se ne può considerare il capostipite. La svolta decisiva, però, ha preso forma solo negli ultimi anni, quando, l'esigenza di strumenti più efficaci, ma utilizzabili anche da soggetti non professionali, ha generato lo strumento chiamato ***wiki***.

La rivoluzione del wiki • Un wiki è uno spazio virtuale condiviso, i cui contenuti sono il frutto della collaborazione e dei contributi di un gruppo o di una comunità i cui componenti intendono condividere tra loro le conoscenze personali.

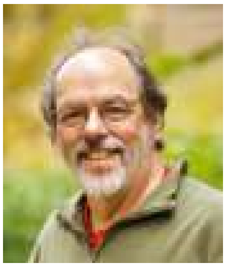


Figura **5**
Ward Cunningham
.

Storia del nome • Lo strano nome di questi contenitori virtuali, "*wiki*", deriva da una termine della lingua hawaiana, "*wiki wiki*", che significa *veloce*, *rapido*. Nell'intento del loro creatore, Ward Cunningham, che li ideò nel 1994, essi dovevano essere strumenti che permettessero lo scambio veloce e immediato di informazioni dei tipi più disparati, in cui i contenuti potessero essere migliorati e arricchiti grazie a molteplici contributi, continuamente aggiornati dai componenti della comunità a cui il wiki fa capo, convergendo verso un livello di correttezza sempre maggiore.

Il concetto fondante era quello di permettere a più autori (***contributors***) di scrivere su un medesimo argomento, ognuno apportando le proprie conoscenze e dando la possibilità anche ad altri, secondo regole precise, di modificare quanto già scritto precedentemente, migliorandolo e correggendolo laddove fosse incompleto, errato o anche solo impreciso.

Nato come strumento di partecipazione attiva, un wiki può anche essere accessibile a utenti che non intendono - o non sono in grado - di contribuire attivamente, ma si configurano come semplici fruitori di contenuti.

Notiamo quindi come, a differenza del blog, se consultassimo a distanza di tempo una medesima pagina di un wiki la potremmo trovare anche completamente mutata, mentre in un blog, il post inserito a una certa data e ora, se ancora disponibile in linea, sarà sempre il medesimo.

Lo strumento • Dal punto di vista dell'architettura, si tratta sempre di un database relazionale complesso, con un'interfaccia utente abbastanza intuitiva da poter essere utilizzata da soggetti aventi una preparazione informatica minima. I wiki possono essere utilizzati su host locali, restando quindi privati all'interno dell'organizzazione per cui sono stati creati, o possono essere resi accessibili all'esterno via rete, attraverso le tecniche della comunicazione via Web.

Anche in questo secondo caso, la loro consultazione e la possibilità di contribuire attivamente sono in alcuni casi vincolate all'autorizzazione di una entità proprietaria del wiki, in altri possono essere del tutto liberi.

In entrambi i casi, l'accesso al sistema è sottoposto al controllo del proprietario del wiki, il quale decide chi può accedere e con quali privilegi (dalla sola lettura fino al controllo totale sul sistema). Alcuni wiki - il più noto è Wikipedia - in modalità di consultazione sono aperti a tutti, mentre in modalità contributor richiedono una registrazione.

Software collaborativo • Parlando in generale, le applicazioni informatiche che permettono a gruppi di utenti di condividere contenuti e che aiutano a organizzare il lavoro comune vengono indicate con il termine di ***software collaborativo***. In quest'ambito i wiki costituiscono la soluzione oggi più diffusa.

Per questo motivo, l'utilizzo in ambito didattico dei wiki è alla base delle *strategie di apprendimento collaborativo*, in cui il materiale di studio, oltre a essere condiviso dal gruppo, è di fatto continuamente creato e arricchito dal gruppo stesso che ne è il fruitore.

Read, Edit, Link, Save • Pur semplificando al massimo il concetto, possiamo dire che le interazioni tra un utente e un wiki sono concettualmente identificabili in queste quattro azioni:

- *read*: leggere (i contenuti del wiki)
- *edit*: editare (cioè modificare) i contenuti
- *link*: collegare le pagine tra loro attraverso elementi ipertestuali. Nelle pagine di un wiki è frequente incontrare link attivi - di colore blu per convenzione - cliccando sui quali si viene rilanciati a pagine contenenti approfondimenti o argomenti correlati
- *save*: salvare la modifica che abbiamo eventualmente apportato

Punti forti, punti deboli • Per comprendere come si possa trarre il massimo vantaggio dai wiki e proteggerli allo stesso tempo, analizzeremo nel prossimo paragrafo la più famosa tra essi, la celebre ***Wikipedia***, che sicuramente abbiamo già molte volte utilizzato durante le nostre ricerche in rete, anche se magari solo come consultatori.

3 Wikipedia

Se stiamo cercando informazioni in rete su un qualsiasi argomento, che si tratti della data della battaglia di Hastings o della ricetta per fare il sushi, molto probabilmente inizieremo facendo una ricerca sul Web.

Se, ad esempio, digitiamo nella finestra di Google:

```
battaglia hastings
```

otterremo un risultato di questo tipo:

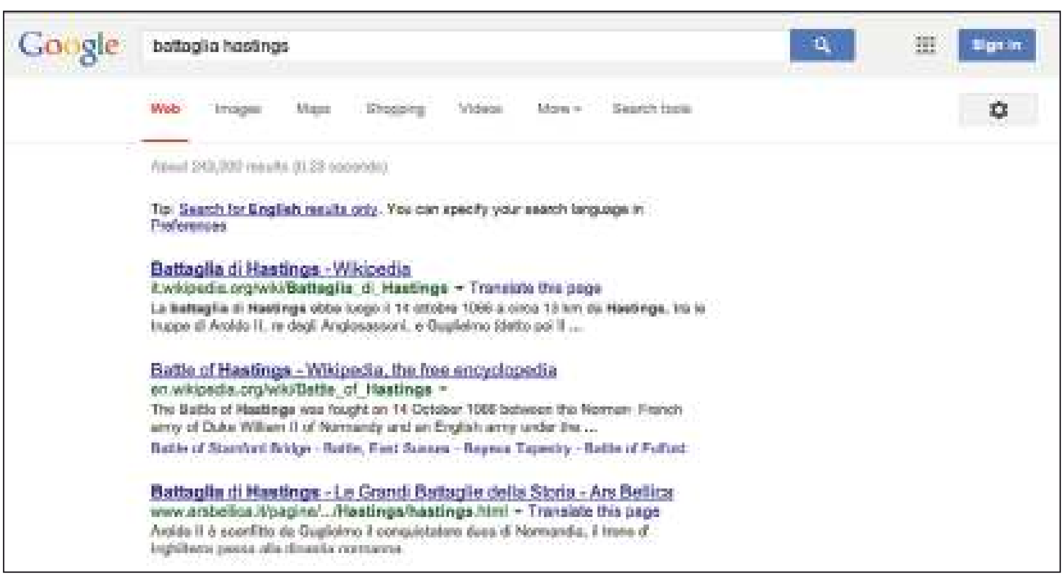


Figura **6**
La risposta di Google alla ricerca "battaglia hastings".

Notiamo - questo capita sempre quando si tratta di argomenti di interesse generale - che, tra quelli proposti, più di uno punta a pagine di Wikipedia. Se scegliamo il primo, visto che rimanda a una pagina in italiano, avremo accesso alla pagina che vediamo nella figura 7.



Figura **7**
La pagina di Wikipedia relativa alla battaglia di Hastings.

Scorrendo la pagina, vedremo che in essa esistono dei contenuti già accessibili in prima lettura, oltre ad alcune immagini, ma anche una serie di link che puntano ad altre pagine, contenenti ulteriori informazioni, tipicamente di approfondimento o inerenti argomenti che sono in relazione con il contenuto della pagina di partenza (figura 8).



Figura **8**
La sezione "Voci correlate" della pagina di Wikipedia.

Ad esempio, se attiviamo il link "Arazzo di Bayeux", troveremo la pagina riportata in figura 9.



Figura **9**
La pagina corrispondente al link "Arazzo di Bayeux".

Quanti wiki? • Abbiamo incontrato una pagina di Wikipedia in italiano: come mai non sono comparse subito le pagine in svedese o in russo? Forse non esistono?
Esistono, molto probabilmente, dato che stiamo parlando di un evento storico fondamentale per la storia europea.

Perché non ci sono apparse per prime? Perché Google, che effettua le ricerche sulla base di algoritmi complessi ed evoluti, quando ha analizzato la nostra stringa di ricerca `battaglia hastings`, ha riconosciuto `battaglia` come una parola della lingua italiana, e quindi ha presupposto, peraltro correttamente, che il nostro idioma di riferimento fosse l'italiano. Se avessimo digitato:

```
bataille Hastings
```

avremmo trovato tra le prime posizioni il rilancio alla pagina in francese di Wikipedia sull'argomento (figura 10).

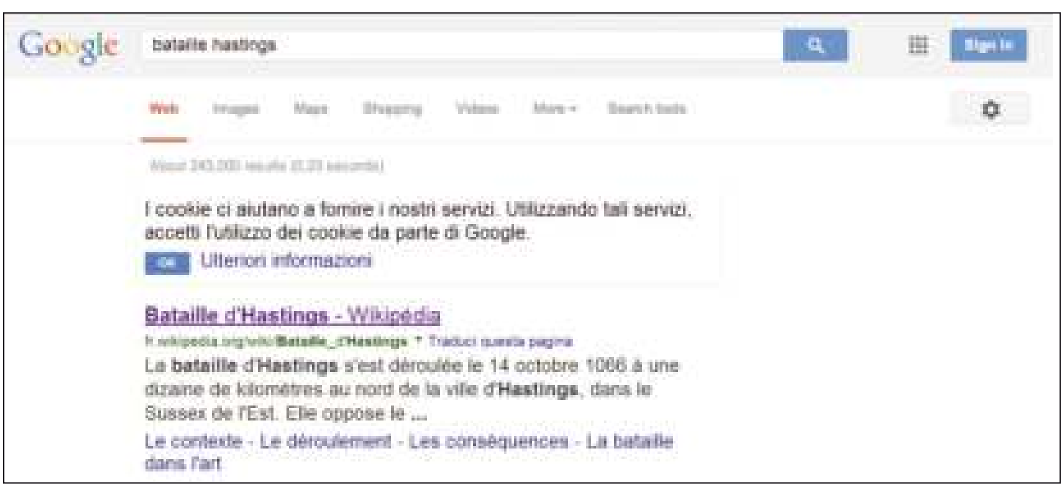


Figura **10**
La risposta di Google alla ricerca "bataille hastings".

Ogni versione di Wikipedia, in ogni lingua, contiene sempre esattamente le stesse informazioni, semplicemente tradotte? Certamente no: Wikipedia è costruita dagli utenti per gli utenti, e i suoi contenuti, come vedremo nel prossimo paragrafo, sono il risultato di quanto i contributors hanno voluto immettere nelle diverse voci. È per questo che una stessa voce può avere descrizioni, link e addirittura informazioni diverse nelle diverse edizioni di quella che è la più grande enciclopedia virtuale del mondo, oppure potrebbe esistere in alcune lingue e non in altre.

Quante lingue? • Al momento della preparazione di questo libro, cioè nel 2014, le lingue in cui erano presenti voci in Wikipedia erano più di 230: in fig. 11 è riportato l'elenco.





Figura **12**
I creatori di Wikipedia: (a) Jimmy Wales e (b) Larry Sanger.
.

Come è nata? • L'idea di utilizzare una struttura di tipo wiki per gestire una enciclopedia virtuale si concretizzò nel 2001 ad opera di Jimmy Wales e Larry Sanger, con la creazione di Wikipedia.
Il nome Wikipedia è la crasi tra la parola *wiki*, di cui abbiamo già parlato, e il suffisso *-pedia* (dal greco *paideia*, che significa *istruzione*). Il successo dell'iniziativa fu immediato, e da allora a oggi, cioè al 2014, le dimensioni di Wikipedia sono divenute veramente planetarie: sono presenti in rete oltre 30 milioni di voci, create e aggiornate da una comunità virtuale di oltre 40 milioni di persone in tutto il mondo. Oltre 10 miliardi di accessi al mese alle sue pagine, fanno di Wikipedia uno dei siti più frequentati nell'universo del Web.

Diventare contributor • Nel consultare Wikipedia, potremmo scoprire che in alcune voci sono presenti degli errori o delle imperfezioni. Ci potrebbe anche capitare di disporre di informazioni che ci piacerebbe mettere a disposizione della comunità aggiungendole alle pagine esistenti: talvolta potremmo essere in grado di creare nuove voci che in quel momento non sono presenti del tutto o in una determinata lingua.

Per diventare contributor, la prima cosa da fare è registrarsi: per fare questo, entriamo nella homepage di Wikipedia nella lingua di nostro interesse, all'indirizzo

```
www.it.wikipedia.org
```

e selezioniamo nella parte superiore della pagina il link che attiva la registrazione (figura 13).



Figura **13**
La pagina home di Wikipedia e il comando "Registrazione".

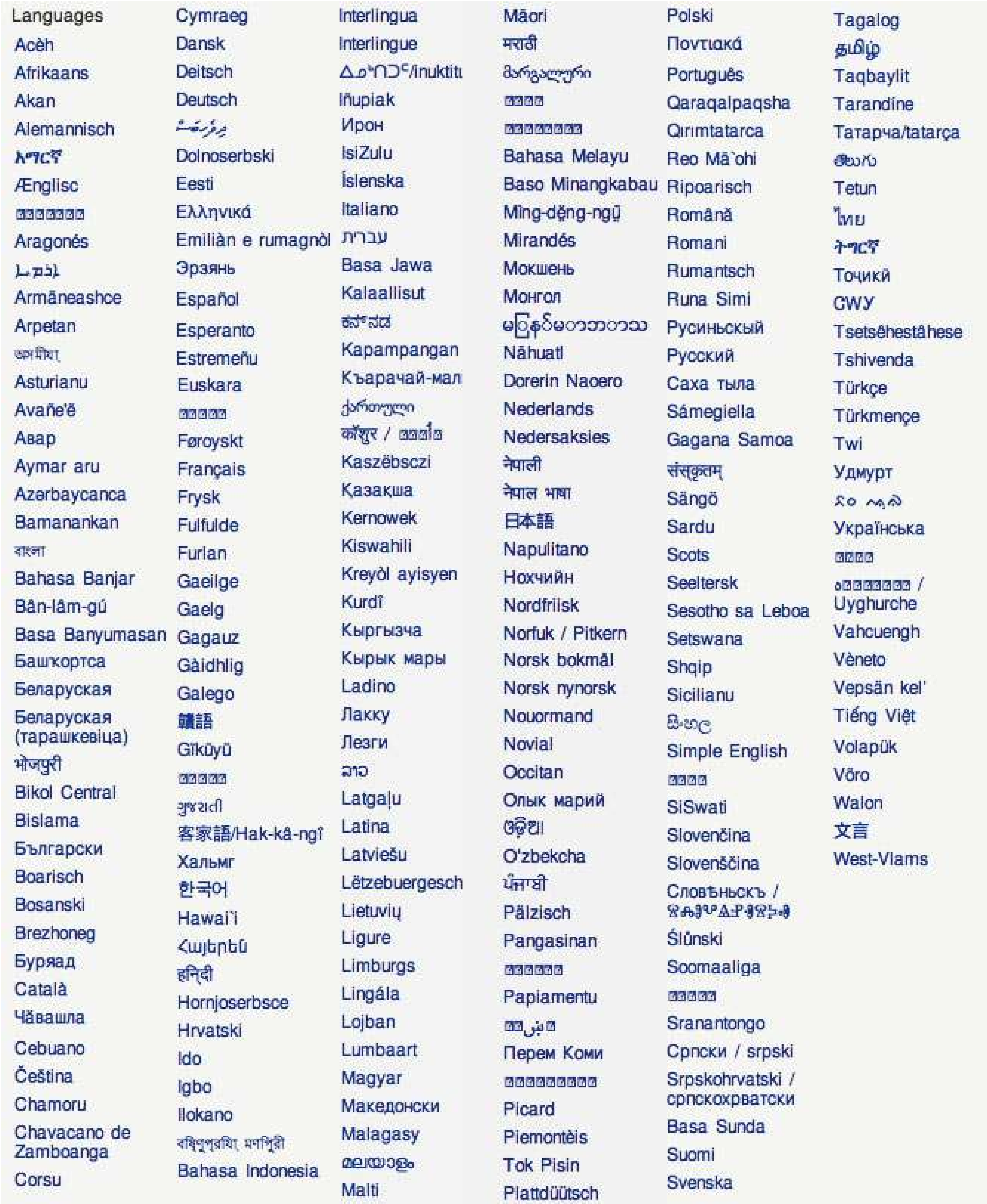


Figura **11:**
Wikipedia: più di 230 lingue diverse!

Il link ci porterà alla pagina di registrazione, nella quale dovremo inserire i nostri dati (figura 14):



Figura **14**
La registrazione in Wikipedia.

Dato che tra gli obiettivi di Wikipedia troviamo quello di avere il maggior numero possibile di autori e di incorporare la massima quantità possibile di informazioni attendibili, nel tempo l'interfaccia utente di Wikipedia è andata via via perfezionandosi, per renderne sempre più semplice e intuitivo l'utilizzo. A questo scopo contribuiscono il sistema di aiuto in linea e la funzione di tutoraggio, attraverso la quale il neofita che vuole diventare un contributor può chiedere di essere affiancato, per dubbi e domande, da un utente più esperto.

I cinque pilastri • La partecipazione a Wikipedia in qualità di contributor è subordinata all'accettazione di cinque principi fondamentali, detti in italiano "*pilastri*", cioè colonne portanti. Li riportiamo integralmente qui di seguito.

Figura **15**
Il logo di Wikipedia
.

Wikipedia è un'enciclopedia che comprende caratteristiche delle enciclopedie "generaliste", delle enciclopedie "specialistiche" e degli almanacchi. Wikipedia non è una raccolta indiscriminata di informazioni. Non è una fonte primaria, ma piuttosto uno strumento di divulgazione secondaria e terziaria; non è un dizionario, né un palco per comizi, né un giornale; non è neanche un luogo nel quale fare promozione, né un banco di prova per l'anarchia o la democrazia; non è neppure uno spazio web utilizzabile indiscriminatamente, né un posto nel quale inserire le proprie opinioni, esperienze o argomentazioni soggettive; tutti i contributori devono sforzarsi di seguire le politiche comunitariamente basate sulla verificabilità e sul divieto di ricerche originali.

Wikipedia ha un punto di vista neutrale, ovvero le voci non devono contenere l'opinione di una sola parte, ma piuttosto riportare le diverse teorie inerenti all'argomento. Tali teorie devono essere presentate in modo chiaro, imparziale, proporzionale alla loro rilevanza, e con il supporto delle necessarie fonti. Nessuna teoria deve essere presentata come "la migliore" o come "la verità", ma deve essere il più possibile supportata da fonti attendibili, specialmente nelle voci su argomenti controversi. Nel caso sorgessero conflitti circa la versione da ritenersi maggiormente neutrale, è opportuno astenersi da ulteriori modifiche e procedere al suo sviluppo tramite il confronto nella pagina di discussione, seguendo la procedura per la risoluzione dei conflitti, e – nei casi più controversi – bloccando momentaneamente la voce.
Wikipedia è libera, il suo contenuto è quindi modificabile da chiunque seguendo alcuni

codici di condotta. Tutti i testi sono rilasciati secondo le licenze libere Creative Commons Attribuzione-Condividi allo stesso modo (CC BY-SA) e GNU Free Documentation License (GFDL) e possono essere citati o distribuiti rispettandone le prescrizioni. Tenere presente che le voci possono essere modificate da chiunque e non sono mai sotto il controllo di un singolo, nemmeno qualora si tratti del soggetto della voce; di conseguenza ciascuna voce aggiunta o modificata può essere a sua volta modificata liberamente e ridistribuita dalla comunità. Inserire solo materiale compatibile con la licenza CC BY-SA.

Wikipedia ha un codice di condotta: occorre rispettare ciascun wikipediano anche quando non si è d'accordo con lui; ci si comporti civilmente, cercando di prediligere il WikiLove ed evitando conflitti di interesse, attacchi personali o facili generalizzazioni; Wikipedia è un progetto collaborativo: si cerchi il consenso, si evitino inutili "guerre di modifiche", non ricorrendo a ripetuti ripristini delle pagine; si ricordi che ci sono 1 092 915 voci nella Wikipedia in italiano sulle quali lavorare e discutere; si agisca in buona fede senza mai danneggiare Wikipedia per sostenere il proprio punto di vista e presumendo, anche nel valutare il lavoro altrui, la medesima buona fede. Si cerchi di mantenere un atteggiamento il più aperto e pacato possibile nei rapporti fra wikipediani, mostrandosi accoglienti con i nuovi arrivati.

Wikipedia non ha regole fisse, eccetto i cinque principi elencati in questa pagina. Si cerchi dunque di non essere timidi nel modificare le voci, poiché il piacere di contribuire non richiede per forza di raggiungere la perfezione, nonostante questo sia l'obiettivo ultimo dell'enciclopedia. Non ci si preoccupi eccessivamente di fare eventuali pasticci: tutte le versioni precedenti di una voce vengono salvate, per cui è impossibile danneggiare Wikipedia in maniera irreparabile. Ma ci si ricordi, allo stesso modo, che tutto ciò che si scrive sarà conservato per i posteri.

Notiamo che, nel testo riportato, alcuni termini sono in blu e sottolineati (la grafica tipica dei link ipertestuali). Questi termini, nel testo corrispondente che si trova sul sito di Wikipedia rimandano a pagine di chiarimento e/o approfondimento.

Dalla teoria alla pratica • La procedura di iscrizione alla comunità dei contributor di Wikipedia è spiegata in dettaglio e corredata di esempi nell'approfondimento disponibile online.

D1-01
Iscrizione come Contributor

L'economia di Wikipedia • Quanto costa consultare una voce di Wikipedia? Nulla: le informazioni in essa contenute sono messe a disposizione degli utenti in modo del tutto libero e gratuito. Eppure, per mantenere in vita il software necessario per la sua gestione e l'hardware che fa funzionare il sistema, ci sono dei costi non indifferenti: nel 2014, l'organizzazione che ne garantisce il funzionamento impiega come dipendenti interni poco meno di cento persone, che devono comunque essere remunerate.

Esistono poi le voci di spesa relative all'hardware, al software di sistema e alle risorse infrastrutturali ed energetiche necessarie. Tenendo conto che l'organizzazione non può mantenersi in vita grazie ai proventi della pubblicità, che non è presente in Wikipedia, chi si fa carico di questi costi?

Wikipedia sopravvive solo grazie alle offerte volontarie che provengono da tutto il mondo: chi utilizza con vantaggio Wikipedia, spesso per scopi professionali, ha - almeno a livello morale - un debito di riconoscenza verso una realtà che rende disponibili gratuitamente informazioni il cui reperimento attraverso altri canali implicherebbe costi e difficoltà significativi.

Volendo passare dal piano morale a quello materiale, ogni offerta, anche piccola, può aiutare l'organizzazione. Se anche inviare pochi euro ci è difficile, almeno contribuiamo, se in qualche argomento ci sentiamo a buon diritto competenti, a migliorarne i contenuti.
Ricordiamo sempre che la collaborazione è un concetto bidirezionale: non significa solo farsi aiutare dagli altri, ma anche essere noi stessi di aiuto alla comunità.

Concetti essenziali

- Un blog è uno strumento di comunicazione in rete di tipo monodirezionale.
- La comunità virtuale di tutti i blog e delle loro correlazioni prende il nome di blogosphere, in italiano blogosfera.
- Esistono blog amatoriali e blog professionali.
- Se un blog non viene periodicamente aggiornato, tende a perdere di interesse.
- Un wiki è uno spazio virtuale condiviso, l'accesso al quale è regolato dal proprietario del sistema.
- I contenuti di un wiki sono il risultato degli interventi di inserimento e modifica effettuati dalla comunità che vi ha accesso.
- Gli strumenti software che permettono a più persone di costruire una base di conoscenze in cooperazione prendono il nome di software collaborativi.
- Un qualsiasi software collaborativo deve avere delle regole per evitare che regni il disordine e i contenuti siano inaffidabili.
- Nelle strategie di apprendimento collaborativo il materiale di studio è creato e arricchito dai componenti del gruppo che ne è il principale fruitore.
- Wikipedia è la più grande enciclopedia virtuale del mondo e si basa su una tipica architettura wiki.
- È possibile divenire contributor, cioè autore, di Wikipedia, attraverso una semplice operazione di registrazione.
- Chiunque può modificare il contenuto di Wikipedia o creare nuove voci, purché rispetti le regole espresse nei cosiddetti cinque pilastri.

Test

1 Dire se le seguenti affermazioni sono vere o false.
Un blog è essenzialmente uno spazio virtuale in rete per a comunicazione di tipo:

A nazionale V F
B bidirezionale V F
C monodirezionale V F
D transnazionale V F

2 Dire se le seguenti affermazioni sono vere o false.
L'insieme di tutti i blog e delle loro correlazioni prende il nome di:

A blogworld V F
B blogsky V F
C blogwater V F
D blogosphere V F

3 Dire se le seguenti affermazioni sono vere o false.
Se vi sono più autori attivi, lo spazio ideale per la comunicazione si chiama:

A sword V F
B forum V F
C dagger V F
D boolean V F

4 Dire se le seguenti affermazioni sono vere o false.
Il termine wiki deriva da:

A il nome di uno dei suoi autori, Wiki Hendricks V F
B una parola hawaiana V F
C le prime quattro lettere dei soci della ditta che ne brevettò l'idea V F
D una preghiera lappone V F

5 Dire se le seguenti affermazioni sono vere o false.
I programmi che permettono di gestire spazi virtuali a intere comunità di autori ed utilizzatori costituiscono il cosiddetto:

A software collaborativo V F
B software cooperativo V F
C programming space V F
D programming theatre V F

6 Dire se le seguenti affermazioni sono vere o false.
In un wiki, se si è abilitati come contributor, si può:

A aggiungere o modificare contenuti V F
B creare nuove voci V F
C espellere altri contributor V F
D inserire collegamenti ipertestuali V F

7 Dire se le seguenti affermazioni sono vere o false.
I cinque pilastri di Wikipedia rappresentano:

A i 5 argomenti di cui non è consentito parlare V F
B le regole fondamentali per lettori e contributor V F
C le prime 5 voci inserite storicamente in Wikipedia V F
D suggerimenti per l'utilizzo, che non è obbligatorio seguire V F

Esercizi

1 **Dopo aver suddiviso la classe in gruppi, ogni gruppo scelga un argomento di vasto interesse e identifichi in rete almeno due o tre blog in italiano che parlano dell'argomento prescelto.**
Si analizzino lo stile del blog, il pubblico cui è destinato, la frequenza di aggiornamento dei post, l'esistenza o meno di commenti dei lettori e si compili una breve relazione di analisi su ognuno di essi.

2 **Utilizzando gli strumenti gratuiti disponibili in rete, quali ad esempio Wordpress, si crei un blog di classe in lingua italiana e lo si mantenga aggiornato, scegliendo un tema sul quale coinvolgere il docente di un'altra materia.**
(http://wordpress.org/)

3 **Utilizzando uno strumento di blog gratuito (vedi esercizio precedente) si crei un blog di classe in lingua straniera (preferibilmente quella prevista nel programma di studio della classe) in cui si discuta una tematica inerente una delle materie scientifiche in studio.**

4 **Si cerchino in Wikipedia notizie in italiano sulla città in cui si trova il proprio istituto scolastico e si esplorino tutte le pagine corrispondenti ai link ipertestuali di primo livello (quelli contenuti nella homepage).**
Si cerchi in Wikipedia se esiste la voce relativa alla propria città in altre lingue: in caso affermativo, si confronti il testo in italiano con quello sullo stesso argomento edito in un'altra lingua straniera che si sia in grado di leggere correttamente; altrimenti, si segua la procedura per diventare contributor e si crei una nuova voce, nella lingua straniera studiata, per descrivere la propria città.
(Esercizio di gruppo, può essere applicato anche ad altri temi a discrezione del docente).

5 **I gruppi dell'esercizio precedente si scambino i ruoli ed entrino in Wikipedia ad analizzare il lavoro di creazione di nuovi argomenti o di approfondimento dei compagni, intervenendo con ampliamenti e, se necessario, con correzioni.**
Il procedimento può essere reiterato con ulteriori interventi, sempre nel rispetto delle regole di Wikipedia. Si analizzi, a distanza di tempo, se sulle voci create dai gruppi sono intervenuti contributi da parte di terzi.
(Esercizio di gruppo, può essere applicato anche ad altri temi a discrezione del docente).

2 Informatica per collaborare

Il capitolo contiene tutte le informazioni essenziali per la creazione di un wiki di classe: grazie all'uso di uno strumento professionale disponibile gratuitamente in rete per scopi didattici, Wikispaces, il docente potrà impostare uno o più wiki insieme agli allievi. A titolo di esempio viene presentato uno spazio virtuale in cui si aggregano molte informazioni, di tipo storico, tecnico, sperimentale, informativo e pratico su un tema condiviso: un'esperienza di verifica sperimentale, nell'ambito del laboratorio di fisica, di una delle leggi fondamentali dell'elettricità. Acquisita la familiarità con lo strumento, il numero dei possibili ambiti di applicazione, anche solo a supporto dell'apprendimento di altre discipline o della gestione di progetti di ricerca in generale diviene, di fatto, illimitato o quasi.

1 Un wiki di classe

Un wiki alla portata di tutti • Abbiamo visto nei paragrafi precedenti come il wiki sia uno strumento altamente versatile: un contenitore all'interno del quale è facile inserire, aggiornare e far crescere le informazioni riguardanti un qualsiasi tema, soprattutto se i contributi vengono da più persone diverse.

Il wiki utilizza la rete: questo permette ai contributor di non dover necessariamente risiedere o lavorare nello stesso luogo, dato che la loro compresenza è virtualizzata attraverso Internet.

Abbiamo anche introdotto il concetto di apprendimento collaborativo: vogliamo quindi presentare uno strumento professionale, utilizzato da milioni di utenti nel mondo, scelto sia per la sua solidità che per il fatto di esistere in versione totalmente gratuita per il cosiddetto scopo *educational* (in italiano, *didattico*): se l'utente che vuole sceglierlo come motore di wiki è il docente di una scuola, per usarlo come strumento di lavoro in classe, lo può fare senza spesa alcuna.

Il wiki che utilizzeremo in questo capitolo e nel seguente, prende il nome di Wikispaces ed è reperibile online all'indirizzo:

www.wikispaces.com

Questo non è il solo motore di wiki gratuito disponibile on line: è in ogni caso quello cui fanno riferimento sia i comandi che le schermate che si troveranno nelle prossime pagine.



Figura **1**
Homepage di Wikispaces

Creare un wiki di classe • Per poter dimostrare lo strumento e al tempo stesso utilizzarlo per un esempio utile dal punto di vista didattico, abbiamo scelto di applicare le nostre conoscenze matematiche e informatiche allo studio della Fisica: nei paragrafi che seguono, imposteremo un wiki di classe al quale parteciperanno docente ed allievi, divisi in gruppi, che svolgeranno una esperienza sperimentale condividendo informazioni, approfondimenti, aggiornamenti e correzioni attraverso il wiki.

Per i nostri scopi, supporremo che il wiki sia già stato creato dal docente, che troverà sul sito associato a questo libro alcune note su come attivare il proprio account, e che l'esperienza su cui ci concentreremo sia nell'ambito dello studio dei fenomeni elettrici. Supporremo la classe suddivisa in tre gruppi, ognuno dei quali parteciperà alla genesi ed all'evoluzione del wiki, intervenendo non solo sulla sezione del wiki di cui dovrà farsi direttamente carico, ma anche su quelle generate dai colleghi.

L'esperienza • Lo scopo, a titolo di puro esempio, di questa esperienza che andremo a svolgere in laboratorio sarà quello di misurare sperimentalmente, cioè attraverso oggetti fisici reali e misure svolte su di essi con appositi strumenti, la resistenza di un conduttore, verificando in questo modo una legge fondamentale relativa ai circuiti elettrici, la prima Legge di Ohm.

Ricordiamo qui brevemente che secondo tale legge la corrente I che scorre attraverso un elemento di resistenza R è direttamente proporzionale alla tensione applicata ai capi dell'elemento stesso. Inoltre, tale proporzionalità presenta un fattore legato alla proprietà dell'elemento di opporsi al transito della corrente, che è espressa dalla sua resistività ρ. Quanto più ρ sarà elevata, tanto meno corrente attraverserà l'elemento a parità di tensione applicata.

Obiettivi • In sintesi, vogliamo parlare della Prima Legge di Ohm, eseguendo una serie di prove pratiche su uno stesso conduttore per arrivare a determinarne il valore di resistenza.

Andremo ad imporre ai capi del conduttore valori diversi di tensione (V) e registreremo l'intensità di corrente I, ricordando che la legge di Ohm vale in condizioni di temperatura costante, per cui tra le grandezze sperimentali che dovremo rilevare comparirà anche la temperatura T alla quale si sono svolti i test.

Se ci trovassimo di fronte a situazioni ideali, in un universo in cui non fossero presenti gli errori di misura, umani o strumentali che siano, la rappresentazione di tutti i dati di tensione e corrente rilevati dovrebbe dare luogo a una retta, passante per lo zero e di coefficiente angolare pari al valore R del conduttore.

Dovremo invece affrontare la realtà delle esperienze nel mondo reale, e comprendere e risolvere, con l'aiuto della Statistica e dell'Informatica, il problema della determinazione della miglior retta che interpoli i dati raccolti, che si troveranno distribuiti sul piano cartesiano molto prossimi ad una retta ma non perfettamente giacenti su di essa.

Impostazione del wiki • Si suppone quindi di impostare il lavoro dividendolo tra i tre gruppi nel seguente modo:

- *Gruppo A*: dovrà occuparsi di eseguire una serie di misure sul conduttore sotto test. Dopo aver messo a punto il circuito elettrico e la strumentazione necessaria, dovrà impostare valori diversi di tensione V e misurare i corrispondenti valori dell'intensità di corrente I.

- *Gruppo B*: partendo dai dati raccolti dal Gruppo A, verificherà se in prima approssimazione i valori di tensione e corrente rilevati possano essere ritenuti linearmente correlati e, con l'ausilio del metodo dei minimi quadrati, calcolerà la regressione lineare per l'insieme dei dati sperimentali, calcolando l'inclinazione della retta, a cui corrisponderà il valore di resistenza cercato

- *Gruppo C*: ricercherà informazioni di tipo storico su Georg Simon Ohm e a quando risalga la scoperta di questa prima legge e delle altre che portano il suo nome; si occuperà

di descrivere la strumentazione utilizzata per l'esperimento e di fornire le informazioni necessarie a chi volesse ripetere l'esperienza nelle stesse condizioni sperimentali

Il docente interverrà per verificare e correggere i contributi.

E' importante, per comprendere al meglio lo scopo di questo tipo di wiki, che essi non siano il semplice contenitore finale dei risultati delle esperienze, ma che vengano usati e fatti evolvere proprio per imparare insieme, attraverso le esperienze dei singoli gruppi, come si giunge ad impostare e risolvere correttamente un problema.

Evoluzione del wiki • Se d'un lato è vero che uno strumento come il wiki permette un'evoluzione, non prevedibile a priori, della struttura delle informazioni in qualche modo correlate ad una determinata voce, o, come si dice in gergo, *entry*, è buona cosa prima di creare un wiki *ex novo* avere almeno un'idea di come lo vorremmo impostare.

Per prima cosa pensiamo al titolo della nostra voce, nel nostro caso potrebbe essere ad esempio "Prima Legge di Ohm" e subito dopo ricordiamo che, per una migliore fruibilità delle informazioni, dovremo evitare d'un lato di voler mettere tutto nella stessa pagina, ma, nel contempo, evitare che la ricerca di una particolare informazione richieda un numero eccessivo di salti e passaggi, via link, da una pagina alla successiva.

Non dobbiamo neppure pensare di creare subito qualcosa di completo e di perfetto: dobbiamo dare la struttura al nostro wiki, cui ogni gruppo andrà ad aggiungere informazioni e considerazioni durante il lavoro sperimentale che ci siamo proposti.

Ogni pagina, ricordiamolo, è sempre editabile e potremo modificarla ed arricchirla, o correggerla, in un secondo tempo tutte le volte che vorremo.

Creiamo un wiki • Per creare la voce, dopo aver digitato le credenziali di autorizzazione all'accesso, basterà cliccare sull'etichetta indicata come *dashboard* (letteralmente, *cruscotto*) in alto a destra, che ci permette di accedere alla seguente pagina (figura 2):

Per creare un nuovo wiki, selezioniamo il link Crea un nuovo wiki ora: ci verrà richiesto, a fini statistici, l'ambito in cui il wiki verrà usato: possiamo indicare *Istruzione Superiore* oppure scegliere di non dichiararlo.

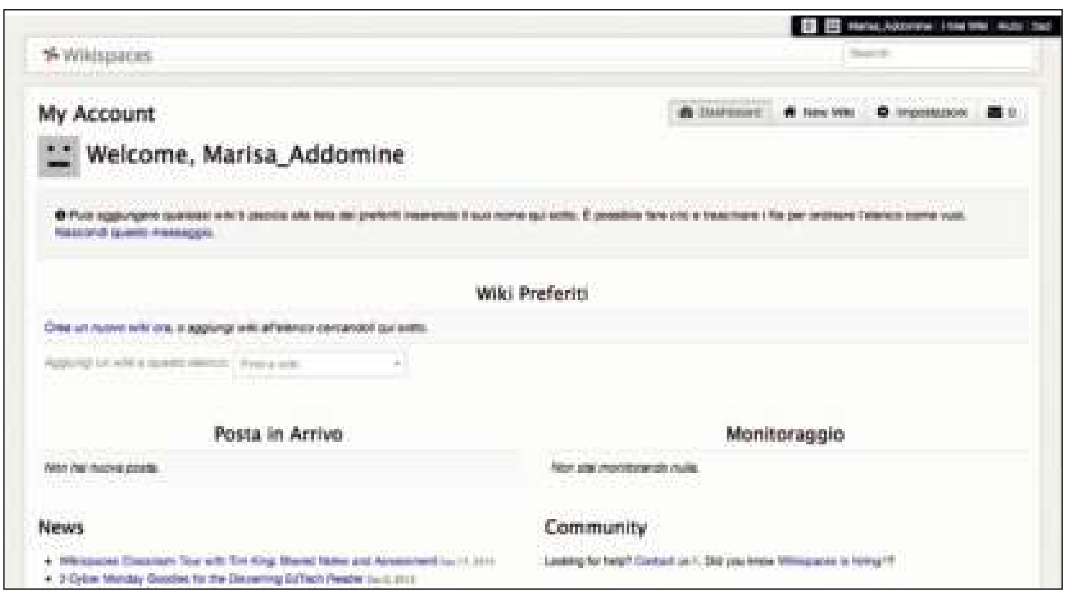


Figura **2**
Pagina di benvenuto al creatore del wiki.

Cliccando su Continua ci troveremo nella pagina di configurazione del wiki (figura 3). Dato che Wikispaces è usato da milioni di utenti, dobbiamo presentarci e compilare con attenzione tutti i campi (i dati che abbiamo utilizzato per l'esempio sono, naturalmente, fittizi).

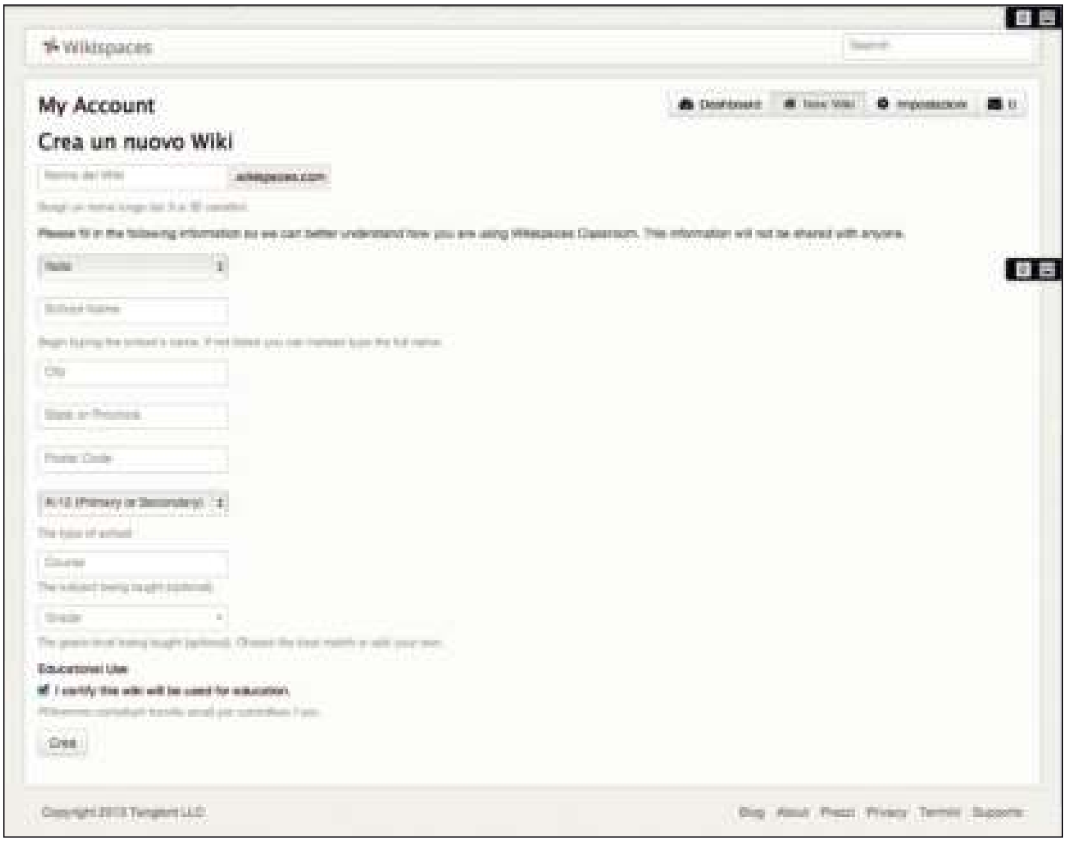


Figura **3**
Il modulo di configurazione del wiki.

In fondo alla pagina troviamo la dichiarazione degli scopi didattici del nostro uso, necessaria per poter accedere gratuitamente a Wikispaces. Terminata la compilazione, cliccheremo sul pulsante Crea (figura 4).



Figura **4**
Il modulo di configurazione del wiki.

Il sistema ci avvisa che *Prima_Legge_di_Ohm* non è un nome consentito, potendo il nome contenere solo lettere o trattini, per cui lo cambieremo in *primaleggediohm*, che verrà accettato.

Stiamo già creando una voce? No, stiamo creando un intero spazio virtuale, all'interno del quale non solo potremo mettere la voce relativa alla Prima Legge di Ohm, ma anche tutte quelle che riterremo di inserire, in funzione dell'utilizzo che anche il docente vorrà suggerire.

Se lo scopo di utilizzo del wiki è più ampio che non lo svolgimento di questo semplice esercizio applicativo, converrà dare un nome più generale al wiki, del tipo *liceoavezzanoquintab*.

Il nostro spazio • Il sistema, dopo qualche istante, presenterà una pagina di benvenuto, indicando che la *classroom*, cioè l'aula virtuale, è stata creata (figura 5).

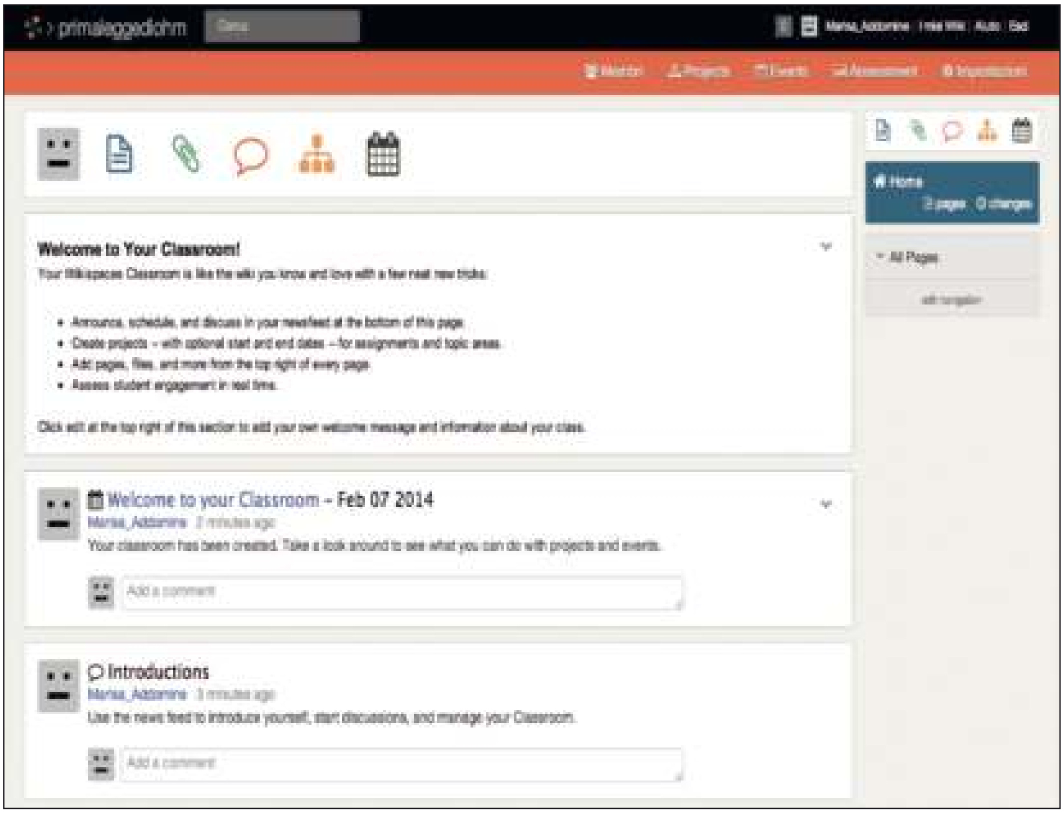


Figura **5**
Pagina a creazione avvenuta.

Notiamo che uno spazio, denominato *Introductions*, in inglese *presentazioni*, ci invita a raccontare qualcosa di noi, come autori di questo wiki.

Creiamo una voce • Siamo ora giunti al punto in cui creeremo la pagina principale dedicata alla voce, quella a partire dalla quale potremo raggiungere tutti i contenuti correlati. Procediamo quindi selezionando nella barra degli strumenti in alto a destra il simbolo della pagina, cliccandolo ed ottenendo:

Digitiamo il nome della nuova pagina, ottenendo la seguente schermata (figura 7):



Figura **6**
Scelta del nome della prima pagina.

Dobbiamo ora inserire il contenuto generale relativo alla voce. Possiamo pensare, tanto per incominciare, di descrivere brevemente di cosa si tratta.

A partire dalla presentazione dovremo poter raggiungere le diverse pagine di cui si dovranno poi far carico i gruppi che si occupano dei vari aspetti.

Non dobbiamo preoccuparci di dare subito una versione "definitiva" ai nostri contenuti, dato che potremo in seguito rieditarli: naturalmente, cercheremo di essere logici ed ordinati per semplificare il lavoro di chi collabora al progetto e rendere più intelligibili, ad ogni livello di approfondimento, i contenuti che inseriremo.

Notiamo che ci viene presentata una sorta di pagina di elaborazione di testi semplificata, con tasti funzione che permettono di agire sull'apparenza del testo (grassetto, sottolineato, ecc.), creare elenchi a punti, ecc. (figura 7).



Figura **7**
Lo spazio per l'inserimento dei contenuti nella pagina.

Supponiamo, ad esempio, di avere iniziato con l'introduzione di un breve testo che rimanda alle varie sottosezioni in cui vogliamo articolare il nostro wiki (figura 8).



Figura **8**
Il testo di base.

Scriviamo il testo, che potremo sempre modificare in un secondo momento.

In ogni momento, mentre stiamo compilando la pagina, possiamo vedere un'anteprima dell'effetto finale utilizzando il tasto Preview : dalla pagina di anteprima potremo sempre ritornare alla pagina di editing.

Una grafica più ricca • Possiamo scegliere di rendere più ricca la nostra pagina sia dal punto di vista estetico, ponendo in risalto alcune parti, inserendo immagini, ecc. ma soprattutto inserendo in essa dei link, cioè dei rinvii ipertestuali ad altre pagine.
Sempre a titolo di esempio, potremmo scegliere di mettere in grassetto la formula V = R x I, ottenendo l'effetto che si può notare nella figura 9.
In ogni momento possiamo salvare la pagina con il tasto Save, per ritrovarla nel nostro spazio ed intervenire su di essa in una ulteriore fase.

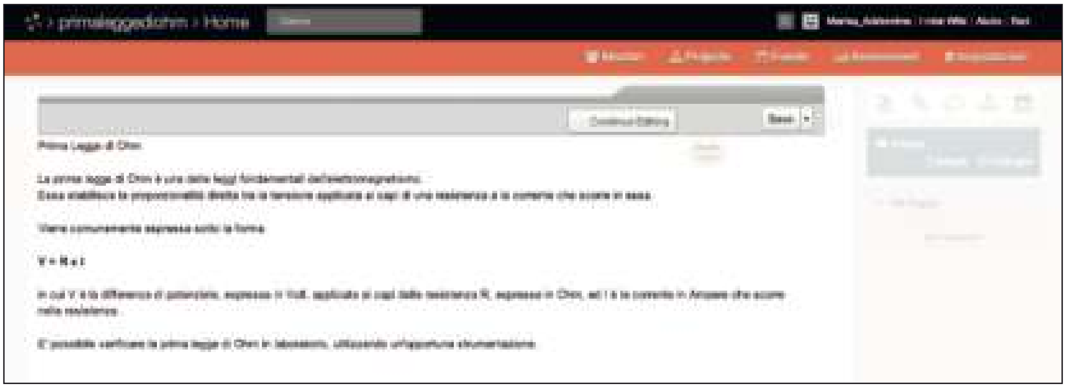


Figura **9**
Testo di base con neretto

Link ipertestuali • Vediamo come inserire un link ipertestuale: ad esempio, vorremmo che alla parola Ohm, intesa come il nome del fisico che formulò questa legge, fosse collegata una pagina di notizie storiche sullo scienziato. Nel testo, selezioniamo la porzione cui vogliamo associare il link e scegliamo il tasto Link nella parte superiore della pagina di editing. Comparirà una finestra di cui verrà richiesta la compilazione (figura 10).

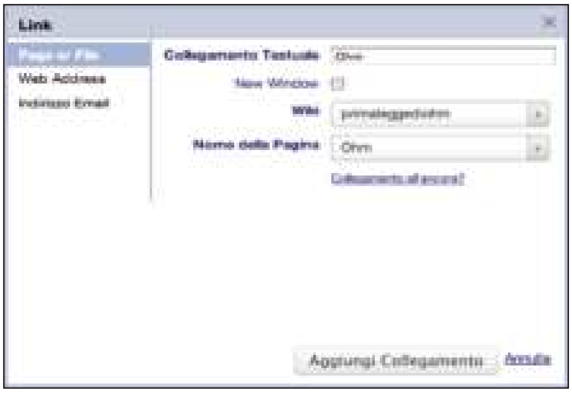


Figura **10**
Modulo per l'inserimento di un link ipertestuale.

Abbiamo collegato un pagina di nome Ohm, che potremo subito anche visitare, modificare o eliminare, se avessimo sbagliato.
Supponiamo ora di salvare e di rivedere quanto fatto finora: comparirà la nostra pagina (figura 11).



Figura **11**
La pagina dopo l'inserimento del link.

Vogliamo provare a usare il link appena creato, sulla parola Ohm: otterremo come risultato l'indicazione che la pagina non è presente.
Notiamo che oltre ad avvisarci che la pagina non esiste, il wiki ci dà la possibilità di editarla:

facciamolo subito, ricordando che potremo poi ampliare o modificare la pagina in un secondo momento.

Per inserire il ritratto di Ohm, che abbiamo trovato sul Web, usiamo il pulsante File, facciamo l'upload dell'immagine, che supponiamo di aver salvato sul nostro PC e che quindi sia disponibile a livello locale, ottenendo quindi il seguente risultato (figura 12):



Figura **12** Inserimento di un'immagine.

Notiamo i punti attivi al contorno dell'immagine, che ci permetteranno di darle la dimensione che vogliamo e di gestire lo scorrimento del testo intorno ad essa.

Salvata la pagina di Ohm, torniamo alla nostra pagina di partenza e verifichiamo che il link ci mandi alla nuova pagina che abbiamo appena creato.

Sulla destra della pagina vediamo un elenco a sfondo grigio che presenta la lista delle pagine: scegliamo quella relativa alla Prima Legge di Ohm e verifichiamo che il collegamento tra le pagine funzioni.

Pochi concetti • Per ragioni di spazio, non seguiremo passo dopo passo la costruzione dell'intera pagina, sulla quale invitiamo docente ed allievi ad esercitarsi. Ricordiamo ancora una volta che il wiki è uno strumento flessibile, per cui, anche se dovessimo commettere degli errori, potremo sempre porvi rimedio.

I concetti fondamentali sono quelli esposti nel paragrafo precedente: esercitandosi e sperimentando si potrà verificare come sia semplice ottenere, con un minimo di pratica, risultati di buon livello.

L'esperimento • Uno dei punti fondamentali del nostro wiki è quello di introdurre, attraverso questo strumento, quello che è il punto focale della nostra esercitazione: lo svolgimento dell'esperimento in laboratorio, attraverso il quale andremo a verificare se la Prima Legge di Ohm è corretta e calcoleremo il valore della resistenza dell'elemento esaminato.

Nel nostro laboratorio di Fisica dovremo quindi avere a disposizione dei materiali con cui costruire il circuito di prova e degli strumenti con cui effettuare le misure. Per ragioni didattiche, forniamo di seguito alcune indicazioni e un insieme di dati misurati, che ci serviranno per ricavare, grazie al linguaggio R, la rappresentazione grafica dell'insieme dei dati e la misura della resistenza a partire dai dati di tensione e corrente.

Strumentazione • Avendo a disposizione un alimentatore in corrente continua stabilizzato, del tipo di quello in figura 13A, con voltmetro e amperometro integrati, collegheremo ai coccodrilli dei cavetti di collegamento i capi del conduttore di cui vogliamo rilevare la resistenza,

una matassina di filo di rame smaltato di lunghezza pari a 10 m e diametro 0,6 mm. Realizzeremo quindi un circuito del tipo riportato in figura 13B.



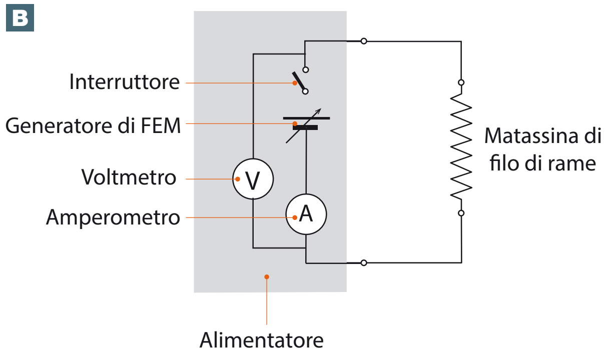


Figura **13**
(A) Alimentatore stabilizzato
(B) Schema del circuito

Partendo da una tensione impostata di 0 V, andremo ad applicare tensioni via via crescenti: possiamo ripetere l'esperimento più volte, sempre raccogliendo i dati con lo stesso metodo, ponendo i risultati delle diverse rilevazioni a confronto.

Tra i dati che forniremo nella pagina apposita, non dimentichiamo di indicare marca e modello dello strumento, oltre che i valori di fondo scala per tensione e corrente, e i valori di precisione delle misure.

Nel nostro caso, supporremo di avere i seguenti dati:

- per le misure di corrente:
  fondo scala: 10A
  sensibilità: ± 0,1A
- per le misure di tensione:
  fondo scala: 30V
  sensibilità: ± 0,1V

Gli errori • Una trattazione esaustiva della teoria degli errori esula dagli scopi di questo testo: è comunque importante ricordare come ogni volta che si effettui una misura il risultato sia sempre e comunque affetto da un errore, per piccolo che sia. Operando in laboratorio, si dovrebbe sempre avere presente il tipo di strumentazione che si utilizza e la precisione che si può ritenere di ottenere per le misure che si stanno effettuando.

I risultati • Supponiamo che al termine della nostra campagna di misure si ottenga la seguente tabella:

Tensione (V)	2,2	3,0	6,1	7,9	10,0	12,6	15,0
Corrente (A)	1,7	2,8	4,8	6,5	6,8	9,3	13,9

Tabella **1**
Risultati sperimentali.

Potremmo pensare di disegnare questi valori sul piano cartesiano a mano, verificando che le coppie tensione - corrente rilevate sono effettivamente poste lungo una linea retta, come afferma la Prima Legge di Ohm: dato che abbiamo a disposizione le nostre conoscenze di Informatica, decidiamo di usare il linguaggio R e di tracciare a video i risultati della nostra esperienza. Impostando il codice nel seguente modo:

```
#
# plottaggio dei valori di tensione e corrente rilevati
# esperimento svolto il 16 febbraio 2014
# Temperatura ambiente: +19°C


tensione=c(2.2, 3.0, 6.1, 7.9, 10.0, 12.6, 15.0)
```

```
corrente=c(1.7, 2.8, 4.8, 6.5, 6.8, 9.3, 13.9)
plot(corrente~tensione, col=2)
```

otterremo a monitor il risultato voluto (figura 14).

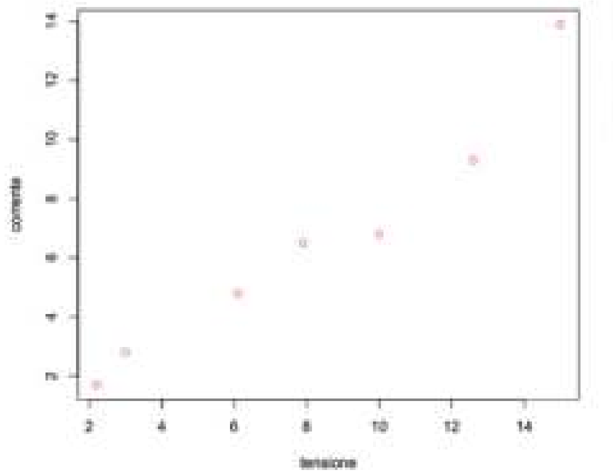


Figura **14**
Il risultato del plottaggio.

Il metodo dei minimi quadrati • È evidente che la distribuzione dei punti sul piano mostra una chiara correlazione di tipo lineare tra i valori di tensione applicata e quelli di corrente misurati. Nel nostro caso, la precisione della strumentazione è buona: lo scostamento dei punti dall'ipotetica retta su cui dovrebbero tutti perfettamente giacere è piccolo.

Un metodo matematico per il reperimento della retta che meglio interpreta la distribuzione quasi lineare di punti intorno ad essa è quello detto dei minimi quadrati: si cerca di ottenere quella retta, che passa attraverso la nuvola di punti in esame, calcolabile facilmente una volta noti il valore medio e la varianza per il campione in esame.

Valore teorico della resistenza del condutture • Per arricchire la nostra esperienza, potremmo pensare di andare a confrontare i risultati delle nostre elaborazioni con quello che dovrebbe essere il valore teorico della resistenza del conduttore in esame, che, per la Seconda Legge di Ohm, dovrebbe essere pari a:

$$R = \rho \frac{l}{S}$$

in cui:
ρ (resistività del rame) = $1{,}7 \cdot 10^{-8}$ Ω/m
l (lunghezza del conduttore) = 10 m
S (sezione del conduttore in m^2): si calcola a partire dal diametro D = 0,6 mm
Lasciamo agli allievi lo svolgimento del semplice calcolo, con cui andranno a confrontare i risultati dell'esperienza in laboratorio.

Dalla teoria alla pratica • Tornando ai nostri dati, cerchiamo una retta che meglio rappresenti tutti i dati raccolti: dovremo quindi cercare quella per cui il residuo, dato dalla somma degli scarti quadratici medi dei dati, è minimo.

Andiamo a impostare con un semplice listato in R il valore medio dei rapporti V/I trovati, per ricavare poi la deviazione standard:

```
#
# partendo dai valori di tensione e corrente rilevati
# si calcolano con R i valori di intercetta e pendenza della
```

```
# curva e la deviazione standard con la funzione sd di R


tensione=c(2.2, 3.0, 6.1, 7.9, 10.0, 12.6, 15.0)
corrente=c(1.7, 2.8, 4.8, 6.5, 6.8, 9.3, 13.9)

# qui impostiamo un modello lineare ed R ci fornisce in uscita
# i valori di intercetta q e di pendenza m della curva
lm(formula = tensione~corrente)

# per avere la deviazione standard sui rapporti V/I
# impostiamo in modo elementare un vettore rapporto
# contenente i rapporti

rapporto=c(2.2/1.7,  3.0/2.8,  6.1/4.8,  7.9/6.5,  10/6.8,  12.6/9.3,
15.0/13.9)

# la deviazione standard e' allora data da
sd(rapporto)
```

Naturalmente, anche il listato del codice che ci ha permesso di effettuare le verifiche dovrebbe essere inserito nel nostro wiki, a disposizione di chiunque volesse ripetere la nostra esperienza o di chi volesse suggerire ulteriori metodi o migliorare la nostra proposta.

Un wiki più ricco • Giunti a questo punto, abbiamo compreso non solo come costruire il nostro wiki per descrivere l'esperimento fatto, ma anche in quanti modi potremmo arricchirlo. Solo a titolo di esempio, si pensi ad introdurre una tabella delle resistività dei materiali più comuni, oppure a scansionare e mettere in linea il manuale di istruzioni dell'alimentatore. Il limite di quanto possa essere collegato al wiki di base è solo dettato dalla nostra fantasia e dal buon senso, che, come sempre, non è, come si dice in gergo, un *optional*.

Concetti essenziali

- Per attivare un wiki è necessario identificarne il gestore, che solitamente è un docente.
- Una volta accreditati, si può subito iniziare a creare un wiki.
- Ogni partecipante al progetto dovrebbe presentarsi, anche concisamente, nello spazio Introductions.
- Ogni pagina del wiki può essere modificata, cancellata, collegata alle altre attraverso link ipertestuali.
- Una volta inseriti i contenuti, si dovrà collaudare il corretto funzionamento dei collegamenti.
- Un wiki è uno strumento totalmente flessibile: è possibile applicarne le potenzialità a moltissimi argomenti, anche in campi del tutto diversi.

Test

1 **Dire se le seguenti affermazioni sono vere o false.**
Lo strumento scelto per creare dei wiki di classe si chiama:

- **A** Wikifront **V F**
- **B** Wikispaces **V F**
- **C** Wikipedia **V F**
- **D** Wikiflash **V F**

2 **Dire se le seguenti affermazioni sono vere o false.**
Quando creiamo una pagina wiki all'interno di un nostro spazio, possiamo:

- **A** darle lo stesso nome di un'altra pagina nello stesso spazio **V F**
- **B** darle un nome in cui compaiano segni di interpunzione e/o spazi **V F**
- **C** darle un nome di fantasia **V F**
- **D** collegarla ad altre pagine, subito o in un secondo momento **V F**

3 **Dire se le seguenti affermazioni sono vere o false.**
In una pagina del nostro wiki possiamo inserire:

- **A** link ad immagini o file esterni **V F**
- **B** codice in linguaggio C che verrà eseguito al momento del caricamento della pagina **V F**
- **C** testi di lunghezza superiore a 160 caratteri **V F**
- **D** solo materiale di cui siamo certi, dato che non sarà possibile reintervenire a modificarlo **V F**

4 **Dire se le seguenti affermazioni sono vere o false.**
Un wiki può essere utilizzato:

- **A** come uno spazio virtuale, in teoria anche per gestire progetti o attività, come contenitore condiviso **V F**
- **B** per controllare un processo industriale **V F**
- **C** come foglio elettronico sempre pronto **V F**
- **D** come un impaginatore professionale **V F**

5 **Dire se le seguenti affermazioni sono vere o false.**
Dati N valori di una grandezza, si definisce media:

- **A** il valore minimo che la grandezza assume **V F**
- **B** il valore ottenuto dalla somma diviso il numero N dei valori **V F**
- **C** il valore massimo che la grandezza assume **V F**
- **D** il valore che almeno il 75% dei campioni assume **V F**

6 **Dire se le seguenti affermazioni sono vere o false.**
Lo scarto quadratico medio è anche detto:

- **A** radice quadrata della media **V F**
- **B** covarianza **V F**
- **C** varianza **V F**
- **D** regressione **V F**

Esercizi

Nota: per tutti gli esercizi si suggerisce un lavoro per gruppi, con il coordinamento e la supervisione del docente. Uno degli obiettivi del capitolo è dimostrare la potenza dei wiki come strumento informatico applicato anche a discipline di altra natura, non necessariamente scientifica. Gli esercizi vogliono quindi essere principalmente una raccolta di spunti, che il docente potrà adattare, ampliare o modificare in funzione degli interessi degli allievi e del coordinamento con i colleghi di altre materie.

1 **Si prenda in considerazione il centro urbano in cui si trova l'Istituto scolastico in cui si tiene il corso e si crei un wiki che, partendo dalle generalità sui terremoti, contenga informazioni sulla storia e sul rischio sismico del luogo. In particolare, si identifichino le scale di misura dei terremoti, lo stato delle conoscenze attuali sulla materia in termini di prevedibilità dei fenomeni e le serie storiche, verificando se esistono eventuali cicli o periodicità.**

2 **Con l'aiuto di un wiki si raccolgano e si discutano informazioni relativamente alle lingue parlate nelle famiglie degli allievi presenti in classe: si colleghino le lingue con informazioni sui Paesi in cui questi idiomi sono parlati e si creino le statistiche relative alla presenza in classe di persone aventi conoscenza di tali lingue. Si descrivano anche gli eventuali alfabeti in cui gli idiomi vengono scritti, se diversi da quello usato per la lingua italiana.**

3 **In collaborazione con il docente di Scienze si analizzino serie storiche relative alle precipitazioni e all'andamento delle temperature nel comune in cui si trova l'Istituto scolastico, presentando in un wiki sia i dati, citando le fonti e le modalità di reperimento degli stessi, che la loro rappresentazione grafica, ottenuta attraverso il linguaggio R. Si analizzino, mese per mese in un periodo di almeno dieci anni, i valori minimi, massimi, medi e la varianza dei valori rilevati.**

4 **Con l'aiuto di un wiki di classe si decida di svolgere un esperimento di Chimica, dedicando le necessarie pagine alla descrizione della strumentazione necessaria, mettendo online i manuali di istruzione dei singoli strumenti ed elaborando i risultati sino ad ottenerne, se applicabile, una rappresentazione grafica oltre che numerica. Si discutano gli eventuali errori di misura e di procedura incontrati nello svolgimento dell'esperimento, che dovrebbe essere ripetuto più volte.**
Se applicabile, si fotografino le varie fasi degli esperimenti e si inseriscano le immagini nelle pagine appropriate.

3 Modelli e popolazioni

Come capitolo finale dell'opera abbiamo ritenuto importante mostrare come gli strumenti informatici siano preziosi alleati nello studio previsionale di eventi o situazioni aventi andamento dinamico nel tempo. A tal fine, vengono dapprima presentati i fenomeni con accrescimento di tipo lineare, in cui a ogni intervallo di tempo l'aumento o il decremento del campione è pari a un valore costante, mentre la seconda parte del capitolo è dedicata ai fenomeni ad andamento esponenziale, che trovano campi di applicazione che spaziano dall'econometria, all'antropologia, alla fisica, alla biologia, alle scienze della terra, per non citare che i più comuni.
Lo strumento informatico, oltre a permettere una semplice rappresentazione grafica dei dati raccolti o calcolati, aiuta il ricercatore a determinare il raggiungimento od il superamento di soglie critiche, quali il limite di crescita di una popolazione in funzione di una risorsa limitata o il tempo necessario affinché una sostanza nociva, in grado di degradare o diluirsi nel tempo, divenga presente nell'unità campione in percentuali inferiori alla soglia considerata accettabile per la salute degli umani o dell'ambiente in generale.

1 Modelli lineari

Se in un sistema di cui sono note le condizioni iniziali si riscontra un incremento o decremento costante della popolazione, vale a dire un aumento o una diminuzione di una quantità fissa K del numero degli elementi dell'insieme in esame, possiamo dedurre che la numerosità della popolazione nel tempo crescerà o decrescerà linearmente.
Vediamolo con un esempio.
Supponiamo che un gruppo sportivo, inizialmente composto da 50 individui, ogni anno accolga 10 nuovi membri, senza che nessuno di coloro che sono già associati se ne vada.
La tabella 1 ci mostra come, al variare del tempo t, espresso in anni, aumenti il numero dei soci del gruppo sportivo:

Tabella **1**
L'andamento nel tempo del numero di soci del club sportivo.

Tempo t (anni)	1	2	3	4	5
Numero dei soci	50	60	70	80	90

Se, attraverso un codice in R, volessimo vedere una rappresentazione grafica dell'andamento del fenomeno, otterremmo:

```
# programma plot lineare crescente

soci <- c(50, 60, 70, 80, 90, 100)
plot(soci)
```

in cui y è il numero dei soci e sull'asse delle ascisse abbiamo il tempo t.
L'andamento della curva (fig. 1) - che nel nostro caso è lineare - ha rappresentazione matematica del tipo:

$$y = mx + q$$

in cui q è il valore di y per $x = 0$, cioè al momento iniziale delle nostre osservazioni.

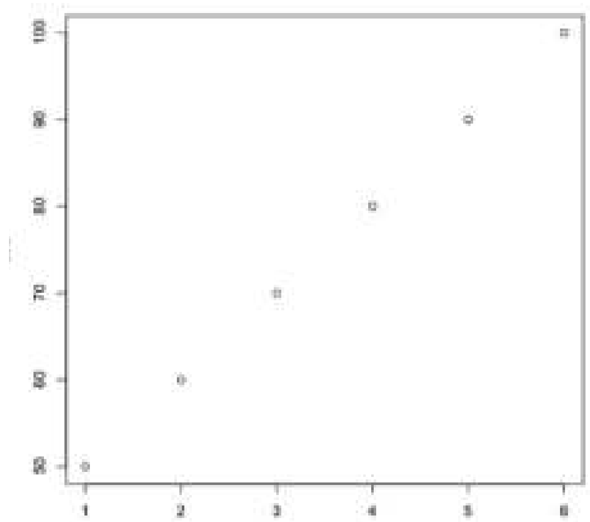


Figura **1**
L'andamento del numero dei soci visualizzato dal codice in R.

In modo del tutto analogo, possiamo considerare un fenomeno di decremento, con perdita di *K* elementi costanti nell'unità di tempo.

Supponiamo di avere un capitale iniziale di 100 000 Euro e di spendere ogni anno 20 000 Euro prelevandoli dal capitale senza in alcun modo reintegrare, in tutto o in parte, il deposito iniziale.

Dalla tabella vediamo facilmente l'andamento del fenomeno:

Tempo *t* (anni)	0	1	2	3	4	5
Capitale (Euro)	100 000	80 000	60 000	40 000	20 000	0

Tabella **2**
L'andamento nel tempo del capitale.

Qui di seguito il codice in R per ottenere una rappresentazione grafica (fig. 2) dell'andamento del fenomeno (una retta con coefficiente m negativo).

```
# programma plot lineare decrescente

capitale <- c(100000, 80000, 60000, 40000, 20000, 0)
plot(capitale)
```

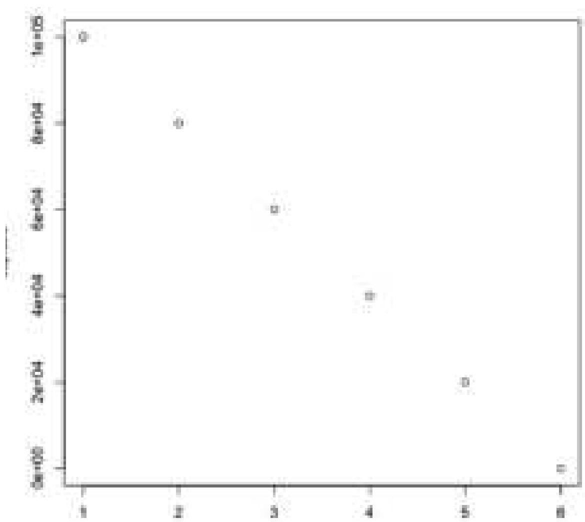


Figura **2**
L'andamento del capitale visualizzato dal codice in R.

2 Modelli esponenziali

Non sempre i fenomeni reali sono modellizzabili con una funzione di tipo lineare: a dire il vero, la maggior parte dei fenomeni nel mondo che ci circonda ha andamenti di tipo non lineare.

Pur con tutte le limitazioni e le approssimazioni che spesso la modellizzazione introduce, esistono correlazioni esprimibili matematicamente in modo molto valido tra x e y, in cui x rappresenta il tempo e y la variabile osservata, che si ritrovano in moltissime situazioni, sia nell'ambito dei fenomeni naturali che in quello delle attività legate alle comunità umane.

Un po' di matematica • Ricordiamo, per comodità. che si dice funzione ad andamento esponenziale una funzione matematica della forma:

$$y(x) = ab^x$$

Nel caso in cui si abbia $a > 0$, se il valore b risulta compreso tra 0 e 1 la curva avrà andamento decrescente (fig. 3A).

Se invece b è maggiore di 1, l'andamento di y sarà crescente al crescere dei valori di x (fig. 3B).

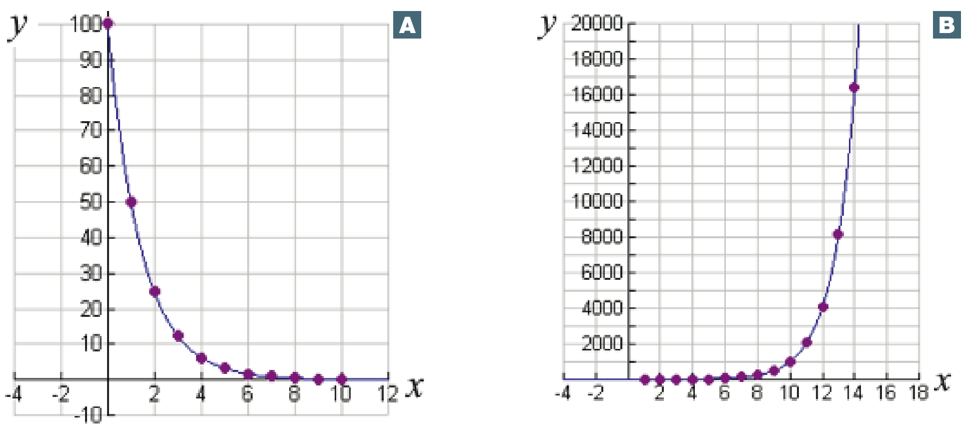


Figura **3**
Andamento esponenziale:
(A) decrescente, con a = 100 e b = 0,5
(B) crescente, con a = 1 e b = 2.

Si potrebbe pensare che queste curve siano solo delle funzioni matematiche, studiate in quanto tali e che esse non abbiano alcun tipo di correlazione con il mondo materiale e le tante situazioni che ci circondano: vedremo invece, attraverso alcuni semplici esempi come in moltissimi casi fenomeni apparentemente scorrelati manifestino nel tempo comportamenti modellizzabili proprio da equazioni di questo tipo.

Crescita e decrescita di una popolazione • Se una quantità y aumenta o diminuisce secondo una percentuale fissa nel tempo, l'equazione generale vista nel paragrafo precedente assume una delle seguenti forme:

$y(x) = a(1+r)^x$ per i fenomeni ad andamento crescente

$y(x) = a(1-r)^x$ per i fenomeni ad andamento decrescente

in cui il significato dei termini è il seguente:

- a è la numerosità del campione al tempo t_0, che coincide con l'inizio delle osservazioni (cioè per $x = 0$)

- r è il tasso di crescita/decrescita della popolazione, tipicamente espresso come percentuale del valore corrente attuale
- x rappresenta la variabile indipendente *tempo*. L'unità di misura scelta dipende dal fenomeno osservato: per le popolazioni umane spesso l'unità di tempo sono gli anni; per i fenomeni di natura biologica possono essere le ore o i minuti, mentre per fenomeni fisici molto rapidi si arriva a usare i sottomultipli dei secondi.

Anche in questo caso, un esempio faciliterà la nostra trattazione.

Il mondo dei batteri • Un caso classico di crescita esponenziale è quello dei batteri che si riproducono, in un ambiente adatto, cioè nel cosiddetto *brodo di coltura* (fig.4), grazie al noto meccanismo di suddivisione cellulare.

A titolo di esempio, esaminiamo la crescita di una immaginaria colonia di batteri il cui numero, inizialmente posto a 1, raddoppi ogni ora. Supponiamo anche che non intervenga alcuna causa di morte o di eliminazione dei batteri dal recipiente in cui li stiamo coltivando.

La crescita della colonia può essere sintetizzata nella seguente tabella:

Tempo *t* (ore)	0	1	2	3	4	5	...	24
Numero di batteri	1	2	4	8	16	32	...	16 777 216

Tabella **3**
La crescita della popolazione di batteri nelle condizioni indicate.

I numeri che esprimono la crescita della popolazione di batteri che ritroviamo indicati nella seconda riga ci dovrebbero essere ben familiari: essi corrispondono, infatti, a:

Tempo *t* (ore)	0	1	2	3	4	5	...	24
Numero di batteri	1	2	4	8	16	32	...	16 777 216
	2^0	2^1	2^2	2^3	2^4	2^5	...	2^{24}

Tabella **4**
La crescita della popolazione di batteri indicata in termini di potenze del 2.

L'espressione matematica di un simile fenomeno di accrescimento è quindi la seguente:

$$y(x) = 2^x$$

Vediamo come si correla alla notazione generale, del tipo

$$y(x) = a(1+r)^x$$

Possiamo immaginarla come scritta nel seguente modo:

$$y(x) = 1(1+1)^x$$

Infatti, $a = 1$ è proprio il numero dei batteri al tempo 0 (l'esperimento è iniziato con un solo esemplare), e il tasso di crescita è 1 (cioè il 100%): infatti, a ogni unità di tempo trascorsa, corrisponde un raddoppio degli elementi presenti nell'insieme.



Figura **4**
La crescita di colonie di batteri diversi in una capsula riempita di brodo di coltura.

Rappresentazione grafica • Impostiamo con R il plottaggio della curva corrispondente ai dati della tabella 3.

```
# programma plot esponenziale crescente

cellule <- c(2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9, 2^10,
2^11, 2^12, 2^13, 2^14, 2^15, 2^16, 2^17, 2^18, 2^19, 2^20, 2^21, 2^22,
2^23, 2^24)
plot(cellule)
```

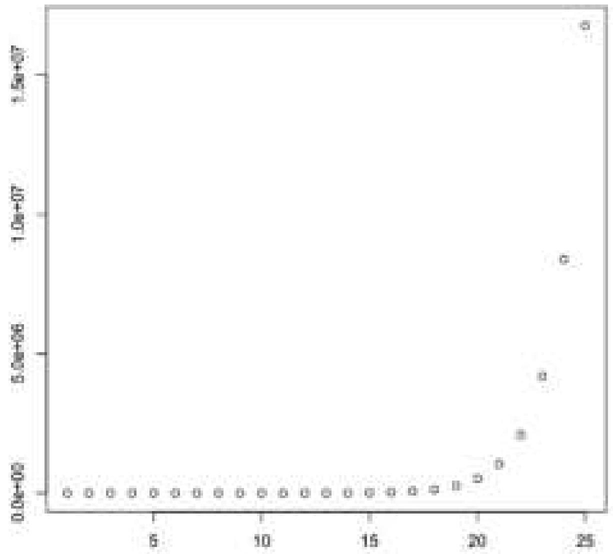


Figura **5**
La crescita della colonia di batteri visualizzato dal codice in R.

Nel grafico riportato in fig. 5 possiamo notare la pendenza con cui le *y* aumentano all'aumentare di *x* anche di poche unità.

In particolare, nel nostro esperimento immaginario, ci accorgiamo che dopo solo 10 ore i batteri sono divenuti

$$2^{10} = 1024$$

e che dopo 24 ore dall'inizio dell'esperimento essi ammontano a

$$2^{24} = 16\,777\,216$$

Limiti dello sviluppo • Si potrebbe quindi pensare che, lasciando la colonia crescere per un tempo abbastanza lungo, ci si potrebbe ritrovare completamente invasi dai batteri, dato il loro tasso di accrescimento. Perché in realtà non è così?

Perché, per questa popolazione come per qualsiasi altro tipo di campione vivente, non si può trascurare l'esistenza di quelle condizioni al contorno che permettono il sostentamento in vita del campione, le cosiddette *risorse*.

Nel nostro esempio, si può trattare del nutrimento, di cui le cellule hanno bisogno: se non è disponibile in quantità illimitata potrebbe non essere più sufficiente ad alimentare l'intera colonia. Potrebbe anche trattarsi di altri tipi di vincoli, quali lo spazio fisico in cui la colonia è contenuta, dato che, se pur molto piccoli, i batteri non hanno dimensione nulla.

Se il campione cui si fa riferimento fosse relativo a specie vegetali oppure agli umani o ad animali superiori, non si potrebbe ignorare che la sopravvivenza del singolo esemplare non è infinita e che esistono dei tassi di mortalità correlati anche a cause diverse dalla semplice morte per invecchiamento, quali le malattie, le morti violente e così via.

3 Applicazioni previsionali

I modelli di tipo esponenziale esprimono molto bene anche i fenomeni in cui la grandezza sotto esame anziché crescere nel tempo vada a diminuire, in funzione di un dato tasso percentuale.

Attraverso tali modelli è possibile prevedere i valori che la grandezza assumerà a un dato tempo *t* oppure quanto tempo il valore di una certa grandezza si abbasserà al di sotto di un certo valore in qualche modo significativo, o lo supererà. Ad esempio, potremmo preoccuparci di quando la concentrazione di una sostanza inquinante scenda sotto la soglia ritenuta sicura per gli abitanti di una città o di quando il numero dei microorganismi necessari per mantenere una fermentazione attiva divenga superiore al valore minimo critico, sotto il quale il fenomeno desiderato non avviene.

Inquinamento ambientale • Un caso tipico è quello dell'analisi del residuo di una sostanza chimica inquinante che si decompone in un terreno. Supponiamo che, per salvaguardare la salute umana e delle specie che vivono in un territorio, una certa sostanza, ad esempio un pesticida, non debba essere presente nel terreno in una percentuale superiore allo 0,1%.

Supponiamo che il pesticida in esame abbia un tempo di dimezzamento noto: cioè che ogni N anni il suo valore residuo nel terreno, sia ridotto al 50% del valore dell'equivalente periodo precedente.

Supponiamo che il produttore del pesticida dichiari che siano necessari 15 anni perché in una zolla di terra la presenza della sostanza chimica in esame dimezzi, e che mediamente in una certa quantità di terra, ad esempio 1000 g si trovino inizialmente 50 g di pesticida.
Dalla tabella 5 vediamo il calo nel tempo della concentrazione di pesticida.

Tempo *t* (anni)	15	30	45	60	75	90	105	120	135
Quantità di pesticida (g/1000 g)	50	25	12,5	6,25	50	1,5625	0,78125	0,390625	0,195312

Tabella **5**
Il calo della concentrazione di pesticida nel tempo.

A questo andamento corrisponde una formulazione analitica del seguente tipo:

$$y(x) = a(1-r)^x$$

in cui:

- un calo corrispondente al 50% (cioè r = 0,5) sulla base di 15 anni. Il valore è compreso tra 0 e 1, quindi è tale da determinare un andamento decrescente.
- a = 100. Infatti per t = 0, cioè all'inizio dell'osservazione, la quantità totale di pesticida risulta pari a 50 g

I due dati risultano evidenti nella tabella 6.

Tempo *t* (anni)	15	30	45	60	75	90	105	120	135
Quantità di pesticida (g/1000 g)	50	25	12,5	6,25	50	1,5625	0,78125	0,390625	0,195312
	$100/2^{10}$	$100/2^{10}$	$100/2^{10}$	$100/2^{10}$	$100/2^{10}$	$100/2^{10}$	$100/2^{10}$	$100/2^{10}$	$100/2^{10}$

Tabella **6**
Il calo della concentrazione di inquinante indicata in termini di potenze del 2.

Vediamo con l'ausilio di R come questi valori siano collocati nello spazio:

```
# programma plot esponenziale decrescente

doseinquinante <- c(50, 25, 12.5, 6.25, 3.125, 1.5625, 0.78125, 0.390625,
0.195312, 0.09765625 )
plot(doseinquinante)
```

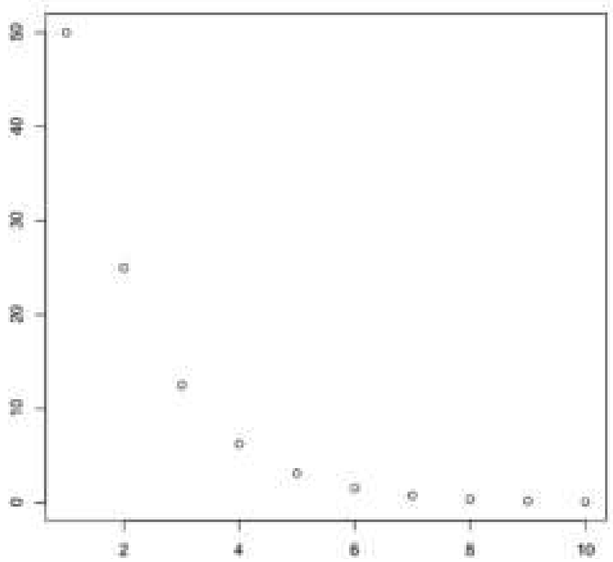


Figura **6**
L'andamento della concentrazione di inquinante visualizzata dal codice in R.

Possiamo verificare come la presenza del pesticida nel terreno scenda al di sotto della soglia di criticità solo dopo più di 90 anni!

Notiamo anche come, con l'andar del tempo, la presenza del pesticida continui a calare ma in modo asintotico, per cui non sarà mai, di fatto, ridotta a zero.

Considerazioni finali • Quando presentato nel capitolo vuol essere solo un punto di partenza e un invito a proseguire, con l'aiuto del docente, l'analisi di fenomeni cui possa essere applicato un modello esponenziale.

Mentre fenomeni lineari presentano valori di crescita costanti in assoluto, quando una popolazione aumenta o diminuisce in forma esponenziale il risultato dell'applicazione del tasso di crescita o di decrescita dà luogo a risultati spesso sorprendenti anche in un numero ridotto di unità temporali.
Si tenga presente, ad esempio, che se una comunità umana o animale ha un tasso di crescita del 7% all'anno, un valore che può sembrare modesto - non importa se la crescita deriva da nuove nascite o dall'arrivo dall'esterno di nuovi membri della popolazione, in assenza di morti o di fuoruscite la popolazione iniziale sarà aumentata del 100% dopo soli 7 anni!

Il punto essenziale, che spesso è trascurato, è il fatto che la percentuale non va a impattare sul solo valore iniziale, ma, a ogni intervallo di tempo, su una popolazione via via crescente, per cui il contributo di crescita in valore assoluto, nell'unità di tempo, è sempre maggiore.

Lo stesso vale, con ovvi adattamenti, per i fenomeni di decrescita esponenziale.

Concetti essenziali

- Se un campione di esemplari cresce (o decresce) di un numero fisso di elementi in uno stesso intervallo di tempo, l'andamento della sua popolazione segue una legge di tipo lineare.

- Un fenomeno che presenta un andamento lineare può essere modellizzato con una funzione del tipo

$$y = mx + q$$

in cui q è la popolazione del campione all'istante t_0 di inizio dell'osservazione.

- Se un campione di esemplari cresce o decresce nell'unità di tempo di una percentuale fissa (calcolata sul numero di elementi del campione al tempo considerato) l'andamento della popolazione è tipo esponenziale.

- La formula generale per un andamento esponenziale è del tipo

$$y(x) = ab^x$$

- La crescita esponenziale di una popolazione può essere espressa più efficacemente con una funzione del tipo

$$y(x) = a(1+r)^x$$

in cui a è il valore della numerosità del campione al tempo x = 0 e r è la percentuale di crescita nell'unità di tempo, con 1 = 100%.

- Se una popolazione ha un andamento esponenziale decrescente, nella funzione

$$y(x) = a(1+r)^x$$

il valore di r è negativo.

Test

1 Dire se le seguenti affermazioni sono vere o false.
La crescita di una popolazione può essere modellizzata attraverso una funzione:

A espansa V F
B contratta V F
C apodittica V F
D stringente V F

2 Dire se le seguenti affermazioni sono vere o false.
Se in un insieme costituito da N elementi iniziali, ad ogni intervallo di tempo T regolare introduciamo M ulteriori elementi, senza mai toglierne, il sistema è di tipo:

A espansivo allineabile V F
B linearmente crescente V F
C logaritmico esponenziale V F
D illimitatamente periodico V F

3 Dire se le seguenti affermazioni sono vere o false.
Nell'espressione

$$y(x) = a\,(1 + r)^x$$

il termine r rappresenta:

A il tasso di crescita o decrescita della popolazione V F
B il valore assoluto del flesso V F
C la numerosità della popolazione in esame V F
D il complemento a 1 di y, espresso in notazione binaria V F

4 Dire se le seguenti affermazioni sono vere o false.
Un sistema in cui il modello di crescita sia del tipo

$$y = mx + q$$

con $m < 0$ sarà:

A strettamente crescente V F
B sinusoidale a tratti V F
C costante nel tempo V F
D linearmente decrescente V F

5 Dire se le seguenti affermazioni sono vere o false.
Un sistema in cui il modello di crescita sia del tipo

$$y = mx + q$$

con $m = 0$ sarà:

A strettamente crescente V F
B sinusoidale a tratti V F
C costante nel tempo V F
D linearmente decrescente V F

6 Dire se le seguenti affermazioni sono vere o false.
Una generica funzione esponenziale ha forma espressa come:

A $xy = k$ V F
B $y = ex$ V F
C $y = e^x$ V F
D $y = a \cdot b^x$ V F

Esercizi

1 **Si consideri una popolazione di 20.000 animali con un tasso di crescita del 10% l'anno, in cui ogni esemplare consumi 500 g di cibo al giorno. Si ipotizzi un ambiente in grado di fornire al massimo 56.000 kg l'anno di cibo, di cui non si riescano a conservare eventuali eccedenze per gli anni a venire. Si calcoli per quanti anni la comunità animale potrà vivere in quel territorio senza avere problemi di approvvigionamento del cibo. Si supponga che gli animali vivano a lungo, quindi che non ci siano decessi durante il periodo in esame. Si illustri graficamente l'andamento del fenomeno con l'aiuto del linguaggio R.**

2 **Il tempo di dimezzamento del carbonio-14 (uno degli isotopi del carbonio) è pari a 5730 anni. Se in un campione che conteneva originariamente 40 grammi di carbonio-14, al momento della prova in laboratorio la quantità di carbonio-14 presente è pari a 15 grammi, approssimativamente, quanto vecchio è il campione? Si rappresenti il risultato graficamente con l'aiuto del linguaggio R.**

3 **Se depositiamo in banca 10.000 euro, sui quali abbiamo un interesse annuo del 4%, non prelevando mai dal conto, dopo 15 anni a quanto ammonteranno i nostri risparmi su quel conto? Si illustri il problema graficamente con il linguaggio R.**

4 **Si modellizzi un sistema in cui una popolazione, inizialmente composta da una coppia di gatti adulti, maschio e femmina, abbia 2 cucciolate l'anno ognuna di 5 piccoli, di cui solo 4 restino in vita per ogni nidiata. Si supponga che la vita dei gatti sopravvissuti sia di venti anni e che non intervengano altre cause di morte accidentale. Ogni anno, i gattini dell'anno precedente, che abbiano compiuto 12 mesi e che si suppongono essere tanti maschi quante femmine, potranno a propria volta riprodursi, sempre al ritmo di 2 cucciolate l'anno. Le risorse del territorio si considerino illimitate. Si calcoli quanti gatti sarebbero in vita dopo 5 anni dall'inizio dell'esperimento.**
Suggerimento: l'intervallo T da considerare è quello tra una cucciolata e l'altra, cioè 6 mesi.

Marisa Addomine Daniele Pons

Informatica

Reti di comunicazione, princìpi di computazione, fondamenti di calcolo numerico

Questo testo presenta in maniera chiara ed efficace i princìpi dell'informatica, accompagnando la teoria con esempi pratici tratti anche da altre discipline scientifiche.

Nel libro

- I concetti di base relativi alle **reti di comunicazione** e una presentazione dell'**architettura Internet a cinque livelli**.
- La teoria della **computabilità** e le applicazioni in ambito informatico della **teoria degli automi**.
- Applicazioni dell'informatica in campi di interesse scientifico: la determinazione dell'integrale definito di una funzione e modelli esponenziali di crescita di una popolazione con il **linguaggio di programmazione open source R**.
- Il web per condividere informazioni e collaborare: **Wikipedia** e **Wikispaces**.
- Alla fine di ogni capitolo si riprendono i **concetti essenziali**, seguiti da **test** per verificare l'apprendimento e da **esercizi**.

Online su www.online.zanichelli.it/addomineinformatica

In questo volume:

- **Schede di approfondimento** (40 pagine) per esempio, *Applicazioni Peer to Peer*, *Algoritmi di routing*

il libro nella nuvola

L'**eBook** sta nella nuvola di Internet e si scarica su tablet, computer e netbook.

- Con le note e i link che aggiunge il professore diventa il **libro della classe**, una piattaforma di collaborazione tra studenti e insegnanti.
- È anche un **quaderno** su cui lo studente scrive appunti e fa esercizi.

L'eBook nella nuvola, leggero e fatto di bit, amplia il libro di carta, contenitore stabile e ordinato del sapere.

NOVITÀ